



**THE PROSPECT OF RESPONSIVE SPACECRAFT USING
AEROASSISTED, TRANS-ATMOSPHERIC MANEUVERS**

DISSERTATION

Robert A. Bettinger, Captain, USAF

AFIT-ENY-DS-14-J-13

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-DS-14-J-13

THE PROSPECT OF RESPONSIVE SPACECRAFT USING
AEROASSISTED, TRANS-ATMOSPHERIC MANEUVERS

DISSERTATION

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Robert A. Bettinger, BS, MA, MS

Captain, USAF

June 2014

DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE;

DISTRIBUTION UNLIMITED

THE PROSPECT OF RESPONSIVE SPACECRAFT USING
AEROASSISTED, TRANS-ATMOSPHERIC MANEUVERS

Robert A. Bettinger, BS, MA, MS
Captain, USAF

Approved:

//signed//
Jonathan T. Black, Ph.D (Chair)

5 May 2014
Date

//signed//
Kerry D. Hicks, Ph.D (Member)

5 May 2014
Date

//signed//
Lt Col Ronald J. Simmons, Ph.D (Member)

5 May 2014
Date

//signed//
Lt Col John R. Dea, Ph.D (Member)

5 May 2014
Date

Accepted:

Adedeji B. Badiru, Ph.D
Dean, Graduate School of Engineering
and Management

Date

Abstract

Comprised of exo- and trans-atmospheric trajectory segments, atmospheric re-entry represents a complex dynamical event which traditionally signals the mission end-of-life for low-Earth orbit (LEO) spacecraft, both manned and unmanned. Transcending this paradigm, atmospheric re-entry can be employed as a means of operational maneuver whereby the aerodynamic forces of the upper atmosphere can be exploited to create an aeroassisted maneuver. Utilizing a notional trans-atmospheric, lifting re-entry vehicle with $L/D = 6$, the first phase of research demonstrates the terrestrial reachability potential for skip entry aeroassisted maneuvers. By overflying a geographically diverse set of sample ground targets, comparative analysis indicates a significant savings in ΔV expenditure for skip entry compared with planar phasing and simple plane change exo-atmospheric maneuvers. In the second phase, the Design of Experiments method of orthogonal arrays provides optimal vehicle and skip entry trajectory designs by employing main effects and Pareto front analysis. Depending on the chosen re-circularization altitude, the coupled optimal design can achieve an inclination change of 19.91 deg with 50-85% less ΔV than a simple plane change. Finally, the third phase introduces the descent-boost aeroassisted maneuver as an alternative to combined Hohmann and bi-elliptic transfers in order to perform LEO injection. Compared with bi-elliptic transfers, simulations demonstrate that a lifting re-entry vehicle with $L/D = 6$ performing a descent-boost maneuver requires 6-12% less ΔV for injection into orbits lower than 650 km. In addition, the third phase also introduces the “Maneuver Performance Number” as a dimensionless means of comparative effectiveness analysis for both exo- and trans-atmospheric maneuvers.

Acknowledgments

I would like to acknowledge and express my sincere gratitude to my research advisor, Dr. Jonathan Black, and the members of my Ph.D Committee – to include Dr. Kerry Hicks, Lt Col Ronald Simmons, Lt Col John Dea, and Lt Col Jeremy Agte – for their invaluable academic guidance and mentorship throughout the research process and composition of this dissertation. I would also like to thank my family, specifically my wife and parents, for their steadfast encouragement and unwavering support, without which my aspirations of attaining my doctorate in Astronautical Engineering would not have come to fruition.

Robert A. Bettinger

Table of Contents

	Page
Abstract	iv
Acknowledgements	v
List of Figures	x
List of Tables	xv
List of Symbols	xvii
I. Introduction	1
General Issue	1
Research Motivation	2
Methodology	4
Preview	12
II. Literature Review	13
Chapter Overview	13
Types of Aeroassisted Maneuvers	13
Aeroassisted Maneuver Performance	14
The Atmospheric Flow Environment and TAV Aerodynamics	19
The Atmospheric Flow Environment and Heat Flux	22
Summary	27
III. Methodology	28
Chapter Overview	28
Assumptions and Limitations	28
Planetary Ellipticity	28
Atmospheric Density and Dynamics	30
TAV Mass Properties	39
Total Force Properties	40
Earth-Based Constants	41
Trajectory Dynamics Model Development	42
Trajectory Dynamics Model Flow Diagram	45
Model Verification Assumptions	46
Verification of Trajectory Dynamics Model	49
Verification of Deceleration Model	60
Verification and Selection of Heat Flux Model	62
Summary and Conclusion	69

IV. Comparative Study of Phasing, Skip Entry, and Simple Plane Change Maneuvers	70
Chapter Overview	70
Introduction.....	70
Methodology.....	72
Simulation of Planar Phasing Maneuvers	72
Simulation of Out-of-Plane Skip Entry Maneuvers.....	83
Simulation of Simple Plane Change Maneuvers	85
Results and Analysis	85
Maneuver Performance Comparison for Select Ground Targets.....	86
Analysis of Out-of-Plane Skip Entry Maneuvers	92
Maneuver Performance Comparison for All Ground Targets	95
Summary and Conclusion	99
V. Design of Experiments Approach to Atmospheric Skip Entry Maneuver Optimization	100
Chapter Overview	100
Introduction.....	100
Methods of Maneuver Optimization.....	102
Methodology	105
Results and Analysis	108
Constant Bank Angle Analysis	108
Variable Bank Angle Analysis.....	118
Single TAV Design Analysis.....	122
TAV Design Application	129
Summary and Conclusion	133
VI. Low Earth Orbit (LEO) Injection and Reachability Utilizing Descent-Boost Maneuvers ..	134
Chapter Overview	134
Introduction.....	134
Maneuver Performance (MP) Number	135
Descent-Boost Maneuver Sensitivity Study	138
Results and Analysis	146
Circular Orbit Injection.....	148
Molniya Orbit Injection	156
Summary and Conclusion	161
VII. Aeroassisted Maneuvers: Potential Air and Space Law Challenges	163
Chapter Overview	163
Introduction.....	163
Applicability of Air and Space Law	164
Spatialism and Aeroassisted Maneuver Altitude Delimitation.....	165
Functionalism and TAV Classification.....	168
Environmental Considerations.....	170
Summary and Conclusion	171

VIII. Conclusions and Recommendations	173
Conclusions of Research.....	173
Significance of Research.....	176
Recommendations for Future Research	177
Appendix A: Exo-Atmospheric Maneuver Algorithms.....	178
Appendix B: Geodesic Equation Formulation.....	183
Appendix C: TLE Guide.....	185
Appendix D: Lambert Algorithm.....	187
Appendix E: MATLAB [®] Code for Trajectory Dynamics Model.....	192
Appendix F: MATLAB [®] Code for Maneuver Simulations	209
Appendix G: MATLAB [®] Code for Support Functions and Utilities.....	312
Appendix H: MATLAB [®] Code for Design of Experiments Support Utilities	331
References.....	385
Vita	394

List of Figures

Figure	Page
1.1. Phasing Maneuver Diagrams: “Ascending” (left) and “Descending” (right)	8
1.2. Simple Plane Change Diagram.....	9
1.3. Hohmann Transfer Diagram.....	9
1.4. Combined Hohmann Transfer Diagram	10
1.5. Bi-Elliptic Transfer Diagram.....	11
3.1. Comparison of Geocentric and Geodetic Latitude	29
3.2. Radial Distance Deviation between Spherical and Oblate Spheroid Models	30
3.3. Initial Comparison of Atmospheric Density Models with MSIS-E-90 and STK [®] Density Data for 01 January 2012	35
3.4. Comparison of MSIS-E-90 and STK [®] Density Data	35
3.5. Comparison of Combined Atmospheric Density Model with MSIS-E-90 and STK [®] Density Data for 01 January 2012	37
3.6. Vehicle Reference Frame and Vector Definition for Sample TAV	40
3.7. Trajectory Dynamics Model Flow Diagram	46
3.8. Bank Angle History for Apollo 10 Command Module Capsule	49
3.9. Comparison of Geocentric/Geodetic Latitude for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)	50
3.10. Comparison of Geodetic Altitude for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)	51
3.11. Comparison of Inertial Velocity for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)	51
3.12. Comparison of Geocentric/Geodetic Latitude for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters).....	52

3.13. Comparison of Geodetic Altitude for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters).....	53
3.14. Comparison of Inertial Velocity for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters).....	53
3.15. Comparison of Geocentric/Geodetic Latitude for Apollo 10 ($C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1$)	55
3.16. Comparison of Geodetic Altitude for Apollo 10 ($C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1$)	56
3.17. Comparison of Inertial Velocity for Apollo 10 ($C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1$)	56
3.18. Comparison of Bank Angle Profile for Apollo 10 ($C_L = 0.40815, C_D = 1.2569, E_{rel} = 1 \times 10^{-8}, N_{max} = \text{Default}$)	57
3.19. Comparison of Bank Angle Profile for $t = [160, 280]$ s ($C_L = 0.40815, C_D = 1.2569, E_{rel} = 1 \times 10^{-8}, N_{max} = \text{Default}$)	58
3.20. Comparison of Geocentric/Geodetic Latitude for Apollo 10 with Non-Interpolation of Bank Angle Profile	59
3.21. Comparison of Geodetic Altitude for Apollo 10 with Non-Interpolation of Bank Angle Profile	59
3.22. Comparison of Inertial Velocity for Apollo 10 with Non-Interpolation of Bank Angle Profile	60
3.23. Comparison of Deceleration for Apollo 10 with Spherical Gravity and Rotating Planetary Model	61
3.24. Pressure Transducer and Calorimeter Locations on the Conical Section of Apollo Spacecraft 009.....	62
3.25. Wing Segment (WS) and Fuselage Section (FS) Locations used for STS-5 Heat Flux Analysis	64
3.26. Re-Entry Trajectory for STS-5	64
3.27. Comparison of Stagnation Heat Flux Models with Flight Data from Sample NASA Vehicles.....	68
4.1. Skip Entry Maneuver Diagram.....	71

4.2. Heading Angle, Orbital Velocity with Respect to a Rotating Reference Frame	75
4.3. Ground Track Trajectory of Reference Orbit ($h_i = 1000$ km, $i_i = 70$ deg).....	77
4.4. Latitude Crossings and Related Longitude Interpolation Solutions	77
4.5. Ground Track Trajectory of “Ascending” Phasing Maneuver Example.....	79
4.6. Ground Track Trajectory of “Descending” Phasing Maneuver Example.....	81
4.7. Maneuver Over-Flight Parameters for Moscow, Russia.....	89
4.8. Maneuver Over-Flight Parameters for Gibraltar, United Kingdom	90
4.9. Maneuver Over-Flight Parameters for Pontianak, Indonesia.....	92
4.10. Over-Flight Detail of Ascending Node Out-of-Plane Skip Maneuver	93
4.11. Over-Flight Detail of Descending Node Out-of-Plane Skip Maneuver.....	94
5.1. Pareto Optimal Front for Campaign #3: $\{\max(\Delta i), \min(\Delta V_{Total})\}$	110
5.2. Pareto Optimal Front for Campaign #3: $\{\max(\Delta i), \max(h_{recirc})\}$	110
5.3. Pareto Optimal Front for Campaign #3: $\{\min(\Delta V), \max(h_{recirc})\}$	111
5.4. Mapping of Pareto Optimal Set from ΔV vs. Δi onto Secondary and Tertiary Objective Spaces	112
5.5. Main Effect on Maximum Inclination Change for DOE Campaign #3 with (a) TAV Mass, (b) Planform Area, (c) Drag Coefficient, and (d) Lift Coefficient.....	113
5.6. Main Effect on Perigee Altitude on Max. Inclination Change for DOE Campaign #3	115
5.7. Augmented Pareto Optimal Front for DOE Campaign #3	116
5.8. Pareto Optimal Front for DOE Campaign #4: $\{\max(\Delta i), \min(\Delta V_{Total})\}$	118
5.9. Main Effect on Maximum Inclination Change for DOE Campaign #4 with (a) TAV Mass, (b) Planform Area, (c) Drag Coefficient, (d) Lift Coefficient	120
5.10. Main Effect on Maximum Inclination Change for DOE Campaign #4 with (a) Perigee Altitude, and (b) Bank Angle.....	121
5.11. Pareto Optimal Front for Single TAV Design: $\{\max(\Delta i), \min(\Delta V_{Total})\}$	122

5.12. Polynomial Fit for Single TAV Design with $\Delta V = f(\sigma)$	123
5.13. Residuals Plot of Polynomial Fit for Single TAV Design with $\Delta V = f(\sigma)$	124
5.14. Surface Fit for Single TAV Design with $\Delta i = f(\sigma, \Delta V)$	125
5.15. Residuals Plot of Surface Fit for Single TAV Design with $\Delta i = f(\sigma, \Delta V)$	126
5.16. Three-Dimensional Solution for Single TAV Design with $\sigma \in [-120, 0]$ deg, $\Delta V = f(\sigma)$, and $\Delta i = f(\sigma, \Delta V)$	127
5.17. Pareto Optimal Front for Single TAV Design: $\{\max(\Delta i), \max(h_{recirc})\}$	128
5.18. Pareto Optimal Fronts for Single TAV Design with (a) Re-Circularization at Skip Apogee, (b) Re-Circularization at $h = 500$ km via Hohmann Transfer	129
5.19. Reference Orbit and Perturbed Orbit Ground Track Trajectories of Single TAV Design	130
5.20. Altitude Profile for Perturbed Orbit of Single TAV Design.....	131
6.1. Descent-Boost Maneuver Diagram	135
6.2. Descent-Boost Apogee Altitude with Variable Initial Altitude and Boost Impulse.....	139
6.3. Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) ΔV vs. γ_i , and (b) Δi vs. γ_i	142
6.4. Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) h_a vs. γ_i , and (b) h_p vs. γ_i	143
6.5. Comparison of ΔV vs. Apogee Altitude Performance with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Descent-Boost Maneuvers, and (b) Combined Hohmann Transfer Maneuvers.....	144
6.6. Maneuver Performance (MP) Number Analysis for Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg	145
6.7. Maneuver Performance (MP) Number Analysis for Combined Hohmann Transfer Maneuvers with Variable Initial Inclination and $h_i = 2000$ km	145
6.8. Example Circular Orbit Injection via Descent-Boost Maneuver; (a) Truncated Descent-Boost Trajectory with Target Altitude Crossings, and (b) Trajectory with Re-Circularization at $\min(\Delta V_{Inject})$	147

6.9. Three-Dimensional View of Descent-Boost 500 km Circular Orbit Injection with $\gamma_i = -12.5^\circ$, $\Delta V_{Boost} = 0.5$ km/s, $h_i = 1000$ km, $h_p \approx 76$ km, $\sigma = 0$ deg.....	148
6.10. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Descent-Boost Maneuver ΔV , and (b) Bi-Elliptic Transfer ΔV	153
6.11. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg, and $h_i = [1000, 1100, 1200]$ km.....	154
6.12. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Time-of-Flight to Apogee, and (b) h_a vs. h_i for Descent-Boost Maneuvers (Quartic Model, $R^2 = 0.9989$).....	155
6.13. Descent-Boost Maneuver with Molniya Orbit Injection with $\gamma_i = -12.3^\circ$, $\Delta V_{Boost} = 0.5$ km/s, $h_i = 1000$ km, $h_p \approx 78$ km, $\sigma = 0$ deg.....	158
6.14. Three-Dimensional Polar View of Descent-Boost Molniya Orbit Injection	159
6.15. Three-Dimensional Polar View of Descent-Boost Orbit Injection and Molniya 3-42 Orbit Trajectories	160
6.16. Detail of Close-Approach of Descent-Boost Orbit Injection and Molniya 3-42 Orbit Trajectories	161
A.1. Phasing Maneuver Flowchart	182
C.1. Element Mapping for Molniya 3-42 Example TLE	186

List of Tables

Table	Page
1.1. Apollo 10 Re-Entry Initial Conditions	4
1.2. Apollo 10 Command Module Capsule Parameters	5
1.3. Notional Trans-Atmospheric Vehicle (TAV) Parameters	5
3.1. Atmospheric Density Model Parameters.....	36
3.2. RMS Error for Combined Density Model Compared with MSIS-E-90 and STK [®] Density Data	38
3.3. Earth-Based Constants	41
3.4. RMS Errors for Modifications to Trajectory Dynamics Model	48
3.5. RMS Error for Trajectory Dynamics Model Verification.....	54
3.6. RMS Error for Alternate Aerodynamic Coefficients	55
4.1. Geographical Coordinates of Sample Ground Targets of Interest	73
4.2. Reference Orbit Initial States for Over-Flight Analysis.....	73
4.3. Out-of-Plane Skip Maneuver Parameters for Moscow, Russia.....	89
4.4. Simple Plane Change Maneuver Parameters ($h_i = 1000$ km, $i_i = 70$ deg)	91
4.5. Skip Entry and Simple Plane Change Maneuver Comparison ($h_i = 1000$ km, $i_i = 60$ deg)	96
4.6. Skip Entry and Simple Plane Change Maneuver Comparison ($h_i = 1000$ km, $i_i = 70$ deg)	98
5.1. Factors and Associated Level Bounds for TAV Design Parameters.....	107
5.2. Factors and Associated Level Bounds for Supplementary DOE Campaigns	116
5.3. Maneuver Parameters of Augmented Pareto Optimal Front	117
5.4. Maneuver Parameters of Pareto Optimal Front for DOE Campaign #4	119

5.5. Optimal TAV Design and Trajectory	121
5.6. Reference Orbit Initial States for Optimal Design Simulation.....	129
5.7. Perturbed Orbit Initial States for Optimal Design Simulation	130
5.8. Maneuver ΔV Comparison of Orbit Re-Circularization Cases	132
6.1. MP Number Usage Examples with Exo-Atmospheric Maneuvers	137
6.2. Reference Orbit Initial States for Descent-Boost Simulations	138
6.3. Trajectory Parameters for Descent-Boost Maneuvers with Variable Boost ΔV at $\sigma = 0$ deg	141
6.4. Comparison of Circular Orbit Injection Performance for Descent-Boost Maneuvers, Combined Hohmann, and Bi-Elliptic Transfers	149
6.5. Initial Flight-Path Angles and Associated Perigee Altitudes for Descent-Boost Maneuvers	151
6.6. Sinusoid Models for Descent-Boost LEO Injection Maneuvers	152
6.7. Comparison of Molniya Orbit Injection Performance for Descent-Boost Maneuver ($\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg), Bi-Elliptic, and Combined Hohmann Transfer.....	157
8.1. Optimal TAV Design and Trajectory from DOE Analysis	174
B.1. General TLE and Element Description.....	185
E.1. m-File Classification for Trajectory Dynamics Model	192
F.1. m-File Classification for Maneuver Simulations	209
G.1. m-File Classification for Support Functions and Utilities.....	312
H.1. m-File Classification for Design of Experiments Support Utilities	331

List of Symbols

The following list of symbols is alphabetical: Lowercase, then uppercase; Latin, then Greek. Due to the magnitude of distances associated with astrodynamics and re-entry analysis, all of the following symbols containing the base unit of measure of meters (m) are converted to kilometers (km) for all subsequent analysis. For the symbols χ and Δ , the notation subscript (\cdot) indicates an unspecified base unit of measure.

<i>Latin Symbol</i>	<i>Definition</i>	<i>Base Unit of Measure</i>
a	Orbital semi-major axis	m
a_{decel}	Total deceleration	m/s ²
d	General distance	m
e	Orbital eccentricity	unitless
f	Planetary flattening parameter; focal length	unitless; m
g	Gravitational acceleration	m/s ²
h	Altitude	m
i	Inclination angle	rad
m	Vehicle mass	kg
n	Scalar quantity (e.g. number of points)	unitless
β	Maneuver performance (MP) number	unitless
r	Geocentric radial distance	m
t	General time	s
C_D	Coefficient of drag	unitless
C_L	Coefficient of lift	unitless
D	Drag force	kg \cdot m/s ²
E_{rel}	Relative error tolerance	unitless
J_i	Zonal harmonic coefficient (Jeffrey constant)	unitless
L	Lift force	kg \cdot m/s ²
N	Integration step size	s
P	Keplerian orbital period	s
P_n	Legendre polynomial, order n	unitless
\dot{Q}	Heat flux	kW/m ²
RMS	Root mean square	unitless
S	Planform area	m ²
T	Thrust force	kg \cdot m/s ²
V	Velocity	m/s

<i>Greek Symbol</i>	<i>Definition</i>	<i>Base Unit of Measure</i>
α	Atmospheric density parameter	unitless
β	Atmospheric scale height	1/m
γ	Flight-path angle	rad
ε	Specific mechanical energy	m ² /s ²
ϵ	Planetary ellipticity	unitless
θ	Longitude	rad
μ	Gravitational parameter	m ³ /s ²
ρ	Atmospheric density	kg/m ³
σ	Bank angle	rad
φ	Co-latitude	rad
χ	Universal variable	(\cdot)
ψ	Heading angle	rad
$\omega_{(\cdot)}$	Planetary rotation rate	rad/s
Δ	Change in value, i.e. ΔV	(\cdot)
ϕ	Latitude (geocentric)	rad

<i>Symbol Scripting</i>	<i>Definition</i>
() _c	Conditions for circular orbit
() _e	Conditions at entry interface
() _f	Final conditions
() _{gd}	Geodetic value
() _i	Initial conditions
() _j	General index
() _r	Component in radial direction
() _s	Stagnation value
() _v	Component in velocity direction
() _w	Conditions at vehicle surface (wall)
() _L	Component in lift direction
() _{SL}	Conditions at sea-level
() _φ	Component in transverse direction
() ₀	Conditions at a reference radius
() _⊕	Conditions for the Earth
() _∞	Free-stream conditions
^I ()	Measured with respect to an inertial frame
^R ()	Measured with respect to a rotating frame

THE PROSPECT OF RESPONSIVE SPACECRAFT USING AEROASSISTED, TRANS-ATMOSPHERIC MANEUVERS

I. Introduction

General Issue

Traditionally, orbital states and orbit geometry are modified via various maneuvers performed *in vacuo*, such as simple plane changes, combined changes to inclination and/or right ascension of the ascending node (RAAN), and coplanar/non-coplanar phasing. Based on a given mission altitude and the desired change in orbital plane position, however, exo-atmospheric maneuvers have the propensity of becoming prohibitively expensive in terms of ΔV . While ΔV expenditure can be reduced by performing maneuvers at high altitudes or nodal crossings, such options are precluded by mission taskings which seek to maximize inclination change, Δi , while simultaneously minimizing the total maneuver ΔV within a specified time duration. Besides the vacuum of space, the upper atmosphere offers an alternative maneuver environment which primarily has been utilized for re-entry, an event that signals the mission end-of-life for low-Earth orbit (LEO) spacecraft. Departing from this convention, atmospheric re-entry can be employed as a means of operational maneuver whereby the aerodynamic drag of the upper atmosphere is exploited by an entry vehicle to create an aeroassisted, trans-atmospheric maneuver. For the purposes of this research, an entry vehicle represents a subset of spacecraft known as *trans-atmospheric vehicles* (TAVs) that are designed to (1) conduct normal mission functions within LEO, and, (2) operate at hypersonic velocities within the upper atmosphere following a de-orbit maneuver by using lift to complete a specified aeroassisted maneuver and fulfill a specified mission tasking.¹

¹ Daniel Gonzalez, Mel Eisman, Calvin Shipbaugh, Timothy Bonds, and Anh Tuan Le, *Proceedings of the RAND Project AIR FORCE Workshop on Transatmospheric Vehicles* (Santa Monica, CA: RAND Corporation, 1997), 1.

Research Motivation

The attainment of *global reach* is part of a wider *responsive space* initiative within the U.S. Department of Defense and represents a shift from a solution-oriented to a capabilities-oriented approach to space acquisition and space system design, in which the performance of a new system is “intended to respond to new taskings within days, hours or minutes without proscribing how it is done.”² Not restricted to the vacuum environment of space, aeroassisted maneuvers represent an alternative means of achieving global reach and feature the potentiality of changing orbital states and geometry with a lower ΔV expenditure and shorter time-of-flight than conventional exo-atmospheric maneuvers. For the present research, global reach is divided into two categories: (1) *Terrestrial reachability*; and (2) *LEO reachability*. With the first category, terrestrial reachability represents the ability of a TAV to overfly a specified ground target within a fixed operations window by performing an aeroassisted maneuver to change orbit inclination and/or semi-major axis. The second category, LEO reachability, extends the concept of global reach to the LEO altitude regime and represents the ability of a TAV to execute a LEO injection subsequent to an aeroassisted maneuver for the prospect of on-orbit inspection and rendezvous.³

One method for determining the performance potential of aeroassisted maneuvers is through the pursuance of a trajectory-centric analysis approach comprised of either a parametric study or an optimization of the trajectory based on a specified performance index. For both cases, the TAV design is known *a priori* and, in conjunction with the mission tasking, represent the fundamental constraints on aeroassisted maneuver performance. As an alternative, the second

² Robert D. Newberry, “Powered Spaceflight for Responsive Space Systems,” *High Frontier* 1 (2005): 46.

³ NASA defines the upper altitude limit of LEO as 2000 km; National Aeronautics and Space Administration, “Process for Limiting Orbital Debris,” *NASA STD 8719.14A* (Washington, D.C.: National Aeronautics and Space Administration, 2012), 23.

method is optimization-centric and determines performance potential by optimizing the TAV design simultaneously with the maneuver trajectory. Based on a specified set of performance indices within the multiple-objective optimization problem (MOP), aeroassisted maneuver performance becomes the objective space arising from an initial decision space containing not only TAV and trajectory design parameters, but also constraints related to TAV capability, to include available ΔV , maximum deceleration g -loading, and maximum heat flux. Employing these two methodologies, the terrestrial and LEO reachability aspects of global reach will be explored by fulfilling the following research objectives:

- Develop and verify a model for utilizing aeroassisted, trans-atmospheric maneuvers to achieve desired orbital state changes induced by aerodynamic effects. This model will hereafter be referred to as the *trajectory dynamics model*.
- Based on a given TAV design commencing from LEO, determine the terrestrial reachability performance of aeroassisted maneuvers, specifically skip entry, by overflying a series of geographically-separated ground targets at high, medium, and low latitudes. For comparison, planar phasing and simple plane change maneuvers will be simulated as exo-atmospheric alternatives to the aeroassisted maneuvers.
- Employing the Design of Experiments method of orthogonal arrays, determine terrestrial reachability by optimizing the TAV and aeroassisted maneuver designs based the MOP of maximizing orbit inclination change while minimizing total maneuver ΔV . Following optimization, the performance of the TAV and aeroassisted maneuver designs will be compared with that of an exo-atmospheric simple plane change.
- Explore the reachability potential of aeroassisted maneuvers as a means for LEO injection and determine a cursory orbit injection envelope for a TAV commencing from

LEO. Also, provide an assessment of the viability of aeroassisted maneuvers for orbit injection when compared with exo-atmospheric maneuver alternatives, specifically combined Hohmann and bi-elliptic transfers.

Methodology

The trajectory dynamics model produces solutions by integrating a set of six nonlinear, ordinary differential equations of motion which govern the kinetics and kinematics of orbital flight and atmospheric re-entry. As a means of model verification, the Apollo 10 re-entry initial conditions will serve as inputs for the trajectory dynamics model so as to compare the resulting trajectory solutions with the actual re-entry trajectory. In addition to the Apollo 10 capsule parameters, the re-entry initial conditions – expressed as geodetic values with respect to an inertial reference frame – are given in the following tables:

Table 1.1. Apollo 10 Re-Entry Initial Conditions⁴

<i>State</i>	<i>Value</i>
Geodetic Altitude, h_{gd_i}	123.55077 km
Inertial Velocity, ${}^I V_i$	11.06715 km/s
Longitude, θ_i	174.24384 deg E
Geodetic Latitude, ϕ_{gd_i}	23.653003 deg S
Inertial Flight-Path Angle, ${}^I \gamma_i$	-6.6198381 deg
Inertial Heading Angle, ${}^I \psi_i$	18.0683 deg

⁴ Kerry D. Hicks, *Introduction to Astrodynamic Re-Entry*, TR 09-03 (Wright-Patterson AFB, OH: Air Force Institute of Technology, 2009), 377.

Table 1.2. Apollo 10 Command Module Capsule Parameters⁵

Pre-Entry Mass, m	5498.22 kg
Planform Area, S	12.017 m ²
Coefficient of Drag, C_D	0.40815
Coefficient of Lift, C_L	1.2569

Following the verification phase, the trajectory dynamics model is utilized to estimate the terrestrial and LEO reachability envelopes for the skip entry and descent-boost types of aeroassisted maneuvers based on a notional TAV as defined in Table 1.3. Similar to spacecraft such as the X-37B Orbital Transfer Vehicle (OTV) in terms of dimensional area and mass, the notional TAV features a theoretical hypersonic lift-to-drag ratio of $L/D = 6$ that serves to illustrate the trans-atmospheric maneuvering capability of a vehicle with aerodynamic characteristics approaching the maximum of Newtonian flow theory.⁶ By comparison, the hypersonic lift-to-drag ratios of the Space Shuttle and X-33 single-stage-to-orbit concept vehicle are 1.9 and 1.2, respectively.⁷

Table 1.3. Notional Trans-Atmospheric Vehicle (TAV) Parameters

Total Wet Mass, m	5000 kg
Planform Area, S	18 m ²
Coefficient of Drag, C_D	0.5
Coefficient of Lift, C_L	3.0

⁵ Ibid., 379.

⁶ John D. Anderson Jr., *Hypersonic and High-Temperature Gas Dynamics*, Second Edition (Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2006), 52.

⁷ Michael E. Tauber, "Maximum Lift/Drag Ratio of Flat Plates with Bluntness and Skin Friction at Hypersonic Speeds," *NASA TM 88338* (Moffett Field, CA: AMES Research Center, 1986), 3; Kevin J. Murphy, Robert J. Nowak, Richard A. Thompson, and Brian R. Hollis, "X-33 Hypersonic Aerodynamic Characteristics," *Journal of Spacecraft and Rockets* 38, no. 5 (2001): 674.

As shown in Chapter V, terrestrial reachability is also determined by conducting main effects and Pareto front analysis to solve the MOP of maximizing inclination change, Δi , while simultaneously minimizing total ΔV . Implementing the Design of Experiments method of orthogonal arrays, the optimization decision space contains both TAV and trajectory design parameters. Consequently, the notional TAV defined in Table 1.3 represents one combination of vehicle design parameters to be simulated in order to solve the MOP.

Whether skip entry or descent-boost in nature, the aeroassisted maneuvers each commence from a circular reference orbit in the LEO altitude regime. Following a de-orbit burn to transfer from the reference orbit into an elliptical trajectory, the TAV changes the orbital states of inclination and semi-major axis by leveraging aerodynamic forces in the upper atmosphere. The amount of change achievable for the orbital states is a direct function of the trans-atmospheric trajectory perigee altitude as well as the aeroassisted maneuver mechanics, specifically the TAV bank angle and initial velocity. In order to maximize aerodynamic force and, therefore, the reachability potential of the aeroassisted maneuver, the TAV must penetrate deep into the sensible atmosphere during perigee transit at a specified negative bank angle to create a leftward turn based on the prograde motion of the initial reference orbit. While a constant bank angle of $\sigma = -90$ deg is assumed in Chapter IV, the Design of Experiments optimization approach in Chapter V utilizes both a constant and variable bank angle within the orthogonal arrays of experiments. Detailed descriptions of skip entry and descent-boost maneuvers are provided in Chapters IV and VI, respectively.

As a means of evaluating aeroassisted maneuver performance, the following types of atmospheric maneuvers are simulated: (1) Phasing maneuver; (2) simple plane change; (3) Hohmann transfer; (4) combined Hohmann transfer; and (5) bi-elliptic transfer. While other

types of exo-atmospheric maneuver exist, to include planar non-tangential orbit transfers, one-tangent burns, apsides rotations, and Lambert transfers, the present research is restricted to the preceding list.⁸ For the first type of exo-atmospheric maneuver, a circular reference orbit in LEO is simulated for a 24 hour-duration, with the resulting ground track trajectory crossings of the ground target latitude identified and catalogued. If the latitude crossings are to the east of the target, then an “ascending” planar phasing maneuver is formulated so as to create an elliptical, perturbed orbit with both a period and semi-major axis greater than that of the reference orbit. Flight along the “ascending” orbit allows for the Earth to rotate a greater angular distance during the orbit period, thus permitting an over-flight of the target rather than a miss to the east as originally calculated.

With latitude crossings to the west of the target, two options are available to shift the ground track trajectory eastward in order to overfly the target. The first option, a “descending” planar skip maneuver creates an elliptical perturbed orbit with both a period and semi-major axis less than that of the reference orbit. By entering into the “descending” eccentric orbit, over-flight of the target is achieved by traversing a greater angular distance during the orbit period, thus decreasing the westward longitudinal difference to zero. The second option arises when the semi-major axes calculated for a “descending” maneuver are less than the radius of the Earth as a result of a large longitudinal difference between the latitude crossing and target. Although patently infeasible, such cases can be transformed into “ascending” phasing maneuvers which prevent planetary impact at the cost of an increased time-of-flight to target. Both the “ascending” and “descending” phasing maneuvers are shown in Fig. 1.1.

⁸ David A. Vallado, *Fundamentals of Astrodynamics and Applications*, Third Edition (El Segundo, CA: Microcosm Press, 2007), 324, 335, 464.

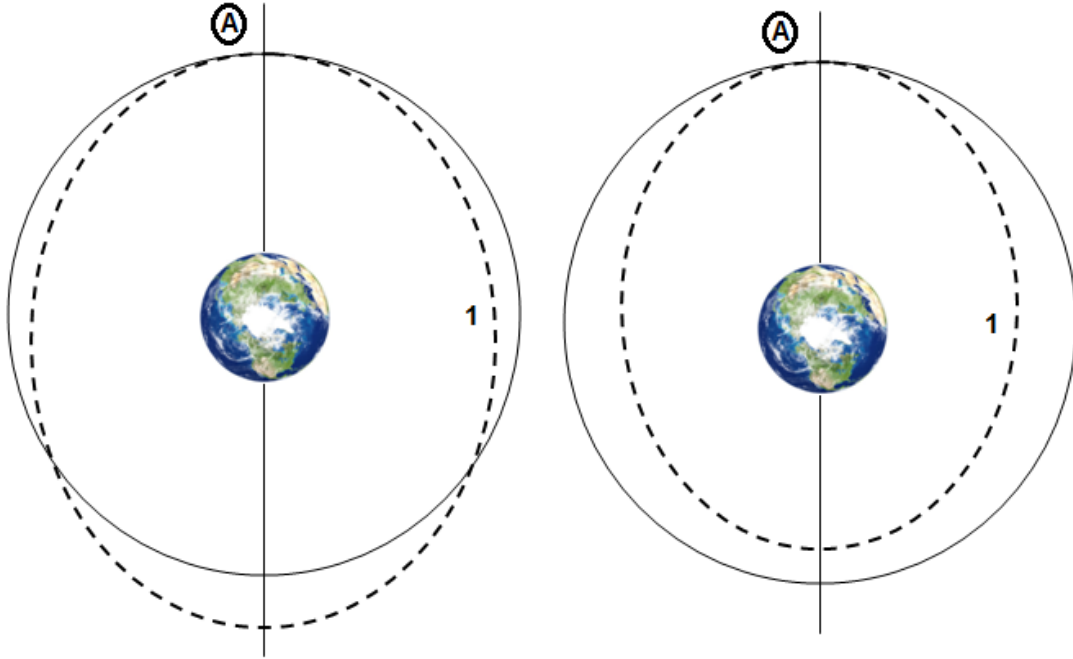


Figure 1.1. Phasing Maneuver Diagrams: “Ascending” (left) and “Descending” (right)

The archetypal out-of-plane exo-atmospheric maneuver, the simple plane change, only creates a change in orbital inclination as ΔV is applied at a nodal crossing. By changing orbital velocity from V_i to V_f , an out-of-plane maneuver is executed which transfers the spacecraft from Orbit (1) to Orbit (2) and thus creating the inclination change Δi as shown in Fig. 1.2. A function of orbital velocity, flight-path angle, and inclination change, an expression for the ΔV necessary to perform a simple plane change is given by:⁹

$$\Delta V_{Simple} = 2V_i \cos \gamma \cdot \sin \left(\frac{1}{2} |\Delta i| \right) \quad (4.7)$$

Known as the *Hohmann transfer*, the second type of maneuver represents one of the most basic and efficient transfer options for altering the orbital semi-major axis. Depicted in Fig. 1.3, the Hohmann transfer is coplanar by definition and consists of a spacecraft first performing a

⁹ Ibid., 345-346.

tangential impulsive burn in circular parking orbit (A) to enter into an elliptical transfer orbit (1) at periapsis. Once in the transfer orbit, the spacecraft does not thrust until apoapsis where another ΔV burn is performed to re-circularize at the desired mission orbit (B).¹⁰

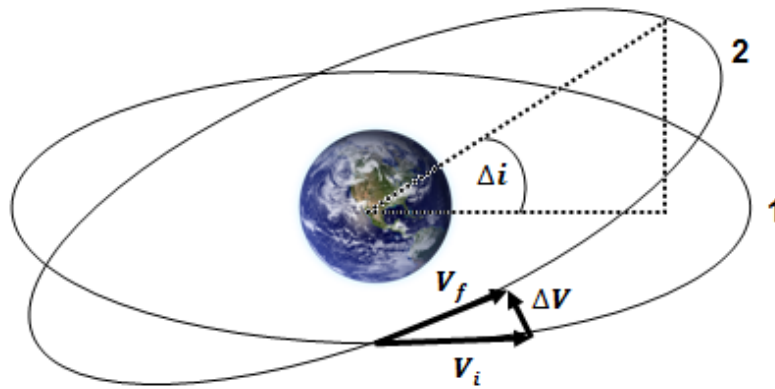


Figure 1.2. Simple Plane Change Diagram

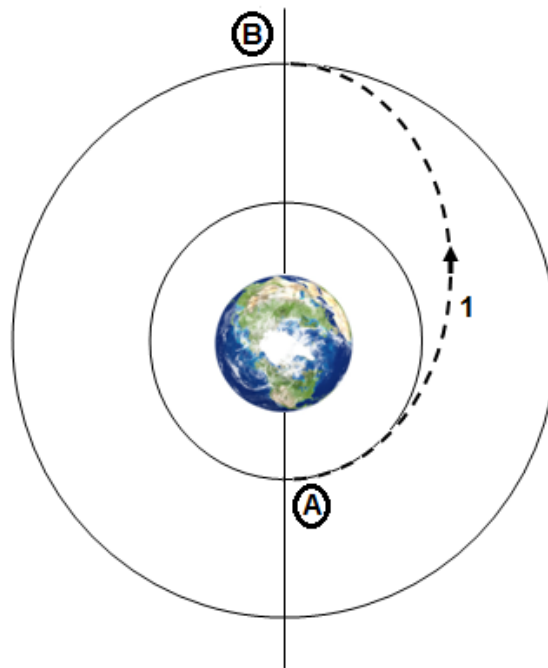


Figure 1.3. Hohmann Transfer Diagram

¹⁰ Robert A. Bettinger and Jonathan T. Black, "Mathematical Relation between the Hohmann Transfer and Continuous-Low Thrust Maneuvers," *Acta Astronautica*, 96 (2014): 42.

For cases in which the parking and mission orbits are non-coplanar, the combined Hohmann transfer in Fig. 1.4 is utilized to change both inclination and semi-major axis. In order to minimize the total ΔV , the inclination change is incorporated into the transfer burns at both (A) and (B) based on the expressions $\Delta i_A = s\Delta i$ and $\Delta i_B = (1 - s)\Delta i$. One option of determining the “best” amount of inclination change to perform at each burn consists of iterating the transcendental equation given by Eq. (1.1):¹¹

$$\sin(\Delta i_A) = \frac{\Delta V_A V_B V_{1,B} \sin(\Delta i_B)}{\Delta V_B V_A V_{1,A}} \quad (1.1)$$

where V_A is the orbital velocity at parking orbit (A), V_B is the orbital velocity at mission orbit (B), $V_{1,A}$ is the velocity at transfer orbit periapsis, and $V_{1,B}$ is the velocity at transfer orbit apoapsis. A second option, which is used for descent-boost maneuver comparative analysis in Chapter VII, involves an analytic approximation that estimates the “best” allocation of inclination change to within about 0.5 deg is shown below, where $R = r_f/r_i$.¹²

$$s \approx \frac{1}{\Delta i} \tan^{-1} \left[\frac{\sin(\Delta i)}{R^{3/2} + \cos(\Delta i)} \right] \quad (1.2)$$

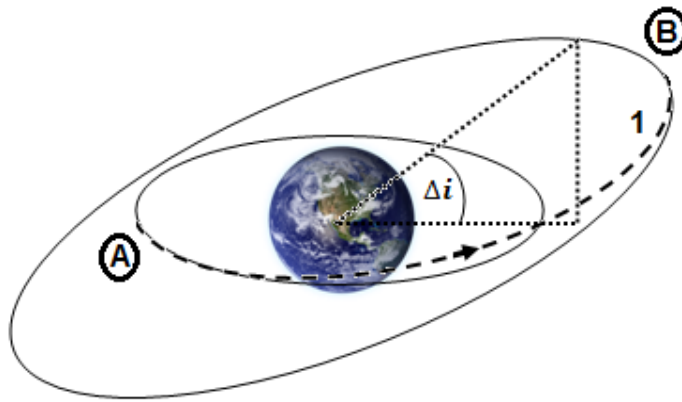


Figure 1.4. Combined Hohmann Transfer Diagram

¹¹ Vallado, 354.

¹² Ibid., 355.

Finally, the bi-elliptic transfer in Fig. 1.5 is similar to the Hohmann transfer such that the parking, mission, and transfer orbits are all coplanar. Although efficient in terms of ΔV , the bi-elliptic transfer features the longest time-of-flight as compared with the preceding maneuvers. Rather than a direct elliptical transfer from the parking to the mission orbit, the bi-elliptic is characterized two transfer ellipses. After performing a tangential impulsive burn at (A), the spacecraft enters into an elliptical transfer orbit (1) until apoapsis at the intermediate orbit (B), which for the example given in Fig. 1.5 is at an altitude greater than the mission orbit altitude. At (B), a second impulsive burn is performed to enter into second elliptical transfer orbit (2) and subsequent re-circularization at the mission orbit (C).

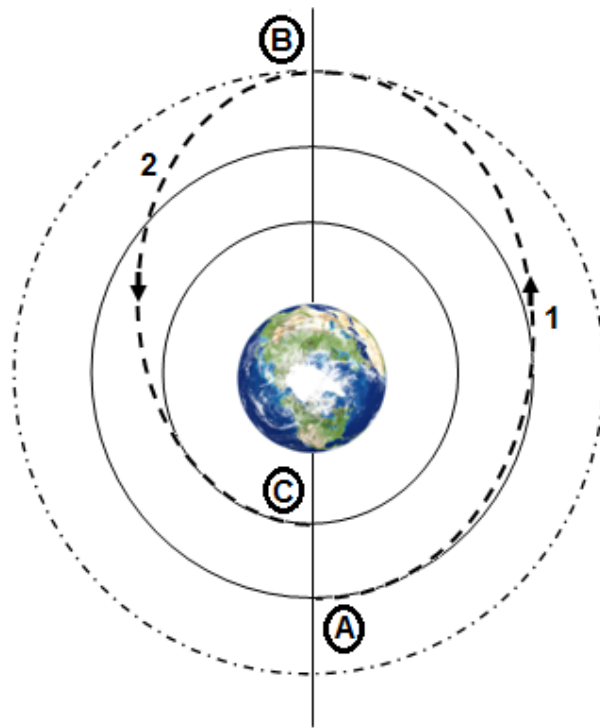


Figure 1.5. Bi-Elliptic Transfer Diagram

Preview

With the research objectives defined and an outline of the analysis methodology provided in Chapter I, Chapter II comprises a review of relevant literature pertaining to aeroassisted maneuvers and the re-entry environment. An extension of Chapter II, a review of literature related to both the Design of Experiments (DOE) method and alternative approaches to maneuver optimization is given in Chapter V. In Chapter III, the first section explores the simplifying assumptions pertaining to the atmospheric density and TAV models which underpin the trajectory dynamics model. The second section provides a detailed presentation of the equations of motion and the gravity model, as well as the verification of the trajectory dynamics, deceleration, and heat flux models. Chapter IV presents a comparative study of ground target over-flight performance for skip entry and exo-atmospheric phasing and simple plane change maneuvers. In Chapter V, the DOE method of orthogonal arrays is employed to optimize both TAV design and the trajectory of an atmospheric skip entry maneuver. Next, Chapter VI examines the use of aeroassisted descent-boost maneuvers for LEO injection and reachability. Chapter VII discusses potential air and space law challenges contemporarily associated with the prospect of executing aeroassisted maneuvers, and, finally, Chapter VIII concludes with a presentation of the significance of the present research as well as recommendations for future research. Presented using the scholarly article format, Chapters IV-VII represent manuscripts submitted to various aerospace engineering journal publications. In terms of ancillary material, Appendix A outlines the algorithms for exo-atmospheric maneuver implementation, Appendix B presents the direct formulation for geodesies on an ellipsoidal planetary model, Appendix C provides a guide for extracting the six Keplerian orbital elements from a Two-Line Element (TLE) set, and an algorithm for solving a Lambert transfer is given in Appendix D.

II. Literature Review

Chapter Overview

The purpose of this chapter is to provide an overview of the relevant research pertaining to aeroassisted, trans-atmospheric maneuvers and their utilization as an alternative to traditional exo-atmospheric maneuvers. Besides analyzing the viability of leveraging aeroassisted maneuvers as a means of altering the orbital elements of a given spacecraft in low-Earth orbit (LEO), preceding studies have also focused on modeling spacecraft aerodynamics as well as the flow and heating environment of the upper atmosphere.

Types of Aeroassisted Maneuvers

Fundamentally, three types of aeroassisted maneuvers can be identified, each representing synergistic maneuvers since they utilize both atmospheric forces – in the form of aerodynamic drag and lift – and propulsive forces. The first type, known as *aerobang* maneuvers, consists of a trans-atmospheric flight trajectory augmented by continuous thrusting at maximum throttle. Employed to not only vary the spacecraft's angle-of-attack, maximum thrust also limits the duration of atmospheric flight, thereby reducing total heating during re-entry. Due to the higher velocity of the aerobang maneuver, however, the spacecraft potentially could experience an increase in re-entry heat flux depending on the altitude of trans-atmospheric flight.¹³ Similarly, the second type of maneuver, known as *aerocruise*, also utilizes propulsive force during the trans-atmospheric trajectory, but at a throttle level sufficient to only counteract aerodynamic drag. The third maneuver type, known as *aeroglide*, is analogous to a skip entry maneuver. Relying primarily on aerodynamic forces, aeroglide maneuvers produce a gliding,

¹³ Richard E. Johnson, "Effects of Thrust Vector Control on the Performance of the Aerobang Orbital Plane Change Maneuver" (MS Thesis, Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, 1993): 3-4.

unpowered trajectory which only employs propulsive forces to de-orbit prior to and re-circularize at the end of the maneuver.¹⁴ Despite experiencing greater total heating stemming from prolonged flight through lower, denser regions of the atmosphere, aeroglide maneuvers are the least expensive in terms of fuel consumption compared with the aerobang and aerocruise alternatives. As measured by the change in orbit inclination per quantity of fuel expended, aerocruise maneuvers have been shown to become increasingly efficient as the bank angle increases during the trans-atmospheric trajectory.¹⁵

Primarily used for interplanetary trajectories, supplementary types of aeroassisted maneuvers consist of *aerobrake*, *aerocapture*, and *aerogravity assist*. Described as purely aerodynamic in nature, aerobrake maneuvers produce a reduction in eccentricity and semi-major axis as a result of aerodynamic drag effects induced with successive perigee passages through the upper atmosphere. Alternatively, aerocapture maneuvers exploit atmospheric drag to reduce orbital energy thereby changing an orbit from hyperbolic to elliptic, while aerogravity assist maneuvers modify the orbital elements of a hyperbolic trajectory by utilizing the combined effects of aerodynamic, gravitational, and propulsive forces.¹⁶

Aeroassisted Maneuver Performance

Skip maneuvers simulated without heat flux path constraints for vehicles in LEO have been demonstrated to have a similar propellant-efficiency with exo-atmospheric maneuvers for changes in inclination less than 3 deg.¹⁷ For $\Delta i > 3$ deg, the propellant expenditure of skip and

¹⁴ Ibid.

¹⁵ John C. Nicholson, "Numerical Optimization of Synergistic Maneuvers" (MS Thesis, Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, 1994), 5.

¹⁶ Christopher L. Darby and Anil V. Rao, "Optimal Impulsive LEO to LEO Multiple-Pass Aeroassisted Orbital Transfer for Small Spacecraft" (paper presented at the 20th AAS/AIAA Space Flight Mechanics Meeting, San Diego, CA, 15-17 February 2010): 3.

¹⁷ Christopher L. Darby and Anil V. Rao, "Minimum-Fuel Low-Earth Orbit Aeroassisted Orbital Transfer of Small Spacecraft," *Journal of Spacecraft and Rockets* 48, no. 4 (2011): 621-622.

simple plane change maneuvers begin to diverge, with simple plane change maneuvers requiring 87% more ΔV to execute a plane change of $\Delta i = 20$ deg. As the inclination change increases to 40 deg, the difference in propellant expenditure also increases with simple plane changes requiring 175% more ΔV than skip entry.¹⁸ Although the minimum maneuver ΔV increases as the number of atmospheric passes increase, skip entry remains more efficient than exo-atmospheric maneuvers for $\Delta i \geq 15$ deg.¹⁹ Even with the imposition of a heat flux path constraint, skip entry maneuvers remain more propellant efficient than exo-atmospheric maneuvers for $\Delta i \geq 15$ deg despite increases in ΔV related to decreases in maximum heat flux.²⁰

In their paper “Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer,” Rao, Tang, and Hallman studied the problem of a minimum-impulse multiple-pass aeroassisted orbital transfer from geostationary orbit (GEO) to LEO with a large inclination change, subject to constraints on heat flux, angle-of-attack, and transfer time.²¹ For their notional TAV, the total aeroassisted inclination change approaches a limit of approximately 36.2 deg as the number of atmospheric passes increases. In all test cases, the aeroassisted maneuver offered “substantial savings” in ΔV when compared with the non-coplanar combined Hohmann and bi-elliptic transfers.²² Similarly, Miele, Lee, and Mease in their paper “Optimal Trajectories for LEO-to-LEO Aeroassisted Orbital Transfer” developed a series of optimal control orbit transfer problems from which to compare the relative performance of aeroassisted maneuvers with that of Hohmann-style, exo-atmospheric maneuvers. Through their analysis, Miele, Lee, and Mease identified that aeroassisted maneuvers required less energy than the bi-elliptic transfer to

¹⁸ Ibid.

¹⁹ Darby and Rao, “Optimal Impulsive,” 45.

²⁰ Ibid., 47.

²¹ Anil V. Rao, Sean Tang, and Wayne P. Hallman, “Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer,” *Optimal Control Applications and Methods* 23 (2002): 215.

²² Ibid., 228-230.

minimize the energy required for orbital transfer, in addition to minimizing the “time integral of the square of the path inclination,” or flight-path angle. For the problem of minimizing the peak heating rate, however, the aeroassisted maneuvers required more energy than the bi-elliptic transfer case.²³

In addition to maneuver comparative analyses, a segment of current literature focuses on the formulation of skip entry guidance algorithms. Specifically tailored for capsule-style entry vehicles with a low lift-to-drag ratio, most of these algorithms provide control guidance during the re-entry phase of a lunar-return trajectory. In their paper “Skip Entry Trajectory Planning and Guidance,” Brunner and Lu developed an on-board, closed-loop numerical predictor-corrector algorithm for re-entry trajectories featuring an initial skip entry flight segment.²⁴ Employing full three-degree-of-freedom dynamics, the algorithm not only computes the required bank angle to achieve the desired final range condition, but also accounts for bank-angle reversals during re-entry, and features lift and drag acceleration filters.²⁵ Intended for use with the Orion capsule, Putnam, Neave, and Barton in “PredGuid Entry Guidance for Orion Return from Low Earth Orbit” formulated a numerical predictor-corrector algorithm that operates a non-spherical planetary model with the inclusion of J_2 -perturbations, and can be used for both lunar and LEO re-entry.²⁶ As an alternative algorithm, Kluever in “Entry Guidance Using Analytical Atmospheric Skip Trajectories” developed a guidance method that uses analytical trajectory

²³ A. Miele, W. Y. Lee, and K. D. Mease, “Optimal Trajectories for LEO-to-LEO Aeroassisted Orbital Transfer,” *Acta Astronautica* 18 (1988): 110, 115.

²⁴ Christopher W. Brunner and Ping Lu, “Skip Entry Trajectory Planning and Guidance,” *Journal of Guidance, Control, and Dynamics* 31, no. 5 (2008): 1210.

²⁵ *Ibid.*, 1218-1219.

²⁶ Zachary R. Putnam, Matthew D. Neave, and Gregg H. Barton, “PredGuid Entry Guidance for Orion Return from Low Earth Orbit” (Paper presented at the *2010 IEEE Aerospace Conference*, Big Sky, Montana, 6-13 March 2010): 2, 6.

solutions obtained from matched asymptotic expansions.²⁷ Further information regarding the mathematical foundation of Kluever's algorithm is found in "Solution of the Exact Equations for Three-Dimensional Atmospheric Entry Using Directly Matched Asymptotic Expansions" by Busemann, Vinh, and Culp,²⁸ as well as "Three-Dimensional Atmospheric Entry Problem Using Method of Matched Asymptotic Expansions" by Naidu.²⁹

Examining the relative performance of aerobang and aerocruise maneuvers in their paper "Optimality of the Heating-Rate-Constrained Aerocruise Maneuver," Ross and Nicholson concluded that the aerobang maneuver is superior to both aerocruise and the exo-atmospheric simple plane change. For the same propellant expenditure, the aerobang maneuver produced an inclination change of approximately 17 deg, whereas the aerocruise and simple plane change alternatives were lower at $\Delta i \approx 15$ deg and $\Delta i \approx 11$ deg, respectively.³⁰ In his paper "Combining Propulsive and Aerodynamic Maneuvers to Achieve Optimal Orbital Transfer," Hanson simulated the synergetic and purely aerodynamic forms of aeroassisted maneuvers and compared the respective orbital transfer performance results with exo-atmospheric maneuvers. Overall, Hanson identified that synergetic aeroassisted maneuvers required the lowest ΔV expenditure by leveraging both aerodynamic and propulsive forces.³¹ Finally, Ikawa and Rudiger in "Synergetic Maneuvering of Winged Spacecraft for Orbital Plane Change" demonstrated that

²⁷ C. A. Kluever, "Entry Guidance Using Analytical Atmospheric Skip Trajectories," *Journal of Guidance, Control, and Dynamics* 31, no. 5 (2008): 1531.

²⁸ Adolf Busemann, Nguyen X. Vinh, and Robert D. Culp, "Solution of the Exact Equations for Three-Dimensional Atmospheric Entry Using Directly Matched Asymptotic Expansions," *NASA CR-2643* (Washington, D.C.: National Aeronautics and Space Administration, 1976): 1-33.

²⁹ D. S. Naidu, "Three-Dimensional Atmospheric Entry Problem Using Method of Matched Asymptotic Expansions," *IEEE Transactions on Aerospace and Electronic Systems* 25, no. 5 (1989): 660-667.

³⁰ I. Michael Ross and John C. Nicholson, "Optimality of the Heating-Rate-Constrained Aerocruise Maneuver," *Journal of Spacecraft and Rockets* 35, no. 3 (1998): 361-364.

³¹ John M. Hanson, "Combining Propulsive and Aerodynamic Maneuvers to Achieve Optimal Orbital Transfer," *Journal of Guidance, Control, and Dynamics* 12, no. 5 (1989): 732-738.

spacecraft performing synergetic aeroassisted maneuvers during high-lift, high-drag flight produce a greater change in inclination than those operating at the maximum lift-to-drag ratio.³²

Performing purely exo-atmospheric maneuvers, Co analyzed the capability of achieving global reach in three separate scenarios: (1) Walker constellation; (2) a single non-maneuvering satellite; and (3) two maneuvering satellites (one with chemical propulsion and the other with electric).³³ In the third scenario, a notional satellite with electric propulsion starting from a 500 km-altitude retrograde orbit performed a series of continuous low-thrusting phasing maneuvers in order to overfly a series of 10 sample ground targets during a 10.5-day campaign. Illustrating the capability of global reach in minimum time, several sample ground targets were overflown, to include Tokyo after an elapsed time of approximately 60 hr with a ΔV expenditure of 0.095 km/s, and Moscow with a time-of-arrival of 140 hr and $\Delta V = 0.18$ km/s.³⁴ Overall, a single electric propulsion satellite was demonstrated to perform a “worst case” of approximately 40 maximum- ΔV maneuvers for a total ΔV of 6.5 km/s. In terms of global reach, it was shown that even the “worst case” targets located furthest from the reference ground track trajectory could be reached and overflown in 2.5 days.³⁵

Extending Co’s research, Dalton in his thesis entitled “Ground Target Over-Flight and Orbital Maneuvering via Aeroassisted Maneuvers” demonstrated the global reach of aeroassisted skip entry maneuvers by identifying terrestrial reachability envelopes for various initial inclination, RAAN, and altitude conditions.³⁶ Assuming both a spherical planetary and

³² H. Ikawa and T. F. Rudiger, “Synergetic Maneuvering of Winged Spacecraft for Orbital Plane Change” (Paper presented at the *AIAA 20th Aerospace Sciences Meeting*, Orlando, FL, 11-14 January 1982): 1-10.

³³ Thomas C. Co, “Operationally Responsive Spacecraft Using Electric Propulsion” (Ph.D Dissertation, School of Engineering and Management, Air Force Institute of Technology (AU), 2012): 218.

³⁴ *Ibid.*, 187-188, 190.

³⁵ *Ibid.*, 226.

³⁶ Devin K. Dalton, “Ground Target Over-Flight and Orbital Maneuvering via Aeroassisted Maneuvers” (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2014): 77-81.

gravitational model, Dalton also developed closed-form analytical equations for the computation of ΔV and time-of-arrival for skip entry, phasing, and simple plane change maneuvers.³⁷

The Atmospheric Flow Environment and TAV Aerodynamics

Underpinning all trajectory analyses and simulations of aeroassisted maneuvers is the method by which the atmosphere is modeled. Due to the short time scales involved with atmospheric entry scenarios, various atmospheric dynamics can be deemed negligible, primarily geomagnetic-induced variations in density and temperature arising due to the solar cycle and related space weather phenomena. As a result, a single atmospheric model can be devised that depicts density as not only decaying exponentially as altitude increases, but also independent of any effects due to time of day, season, or geographic location. Such a model, defined in Vallado's *Fundamentals of Astrodynamics and Applications*, was utilized by Gargasz in his thesis "Optimal Spacecraft Attitude Control Using Aerodynamic Torques," and Hajovsky in his thesis "Satellite Formation Control Using Atmospheric Drag."³⁸

In addition to depicting the macroscopic atmospheric environment as a function of altitude, aeroassisted maneuver simulations have also sought to garner increased model fidelity by capturing the flow characteristics of the upper atmosphere and their relation to TAV aerodynamics. In his study of the viability of achieving three-axis attitude control using only aerodynamic torques, Gargasz divided interactions between the various atmospheric species and a TAV into two categories: specular and diffuse collisions. Storch, in *Aerodynamic Disturbances on Spacecraft in Free-Molecular Flow*, defines specular collisions as deterministic momentum

³⁷ Ibid., 42, 60, 65, 67, 72.

³⁸ Vallado, 562; Michael L. Gargasz, "Optimal Spacecraft Attitude Control Using Aerodynamic Torques" (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2007); Blake B. Hajovsky, "Satellite Formation Control Using Atmospheric Drag" (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2007).

transfer processes in which the angle of incidence equals the angle of reflection, with the incident velocity, reflected velocity, and surface normal all representing coplanar quantities.³⁹ For diffuse collisions, the incident molecules are “trapped into the interstices” of the surface and lose all knowledge of the incoming direction. Subsequently, the molecules are re-emitted from the surface with a random distribution of speed and direction governed by the cosine distribution.⁴⁰

Aside from collisions between atmospheric species and the TAV surface, King-Hele in his book *Satellite Orbits in an Atmosphere: Theory and Applications* identifies a specific example in which interactions with the atmospheric chemical environment directly effects TAV aerodynamics. King-Hele states that while traversing an altitude of 200 – 300 km within the atomic oxygen-rich thermosphere, a TAV acquires “at least a mono-layer” of atomic oxygen on its surface either by mechanisms of chemisorption or physisorption. With this layer present on the TAV surface, most air molecules will strike the atomic oxygen rather than the atoms of the surface material.⁴¹ As a TAV increases altitude above the layer of atomic oxygen and enters the exosphere, atmospheric species predominance shifts from oxygen to helium, and then to hydrogen. King-Hele explains that the decreasing molecular weight of the atmospheric species colliding with the mono-layer of atomic oxygen produces an increase in the TAV drag coefficient from 2.2 to approximately 2.4.⁴²

The flow environment for aeroassisted maneuvers can also be expressed in terms of flow regime rather than momentum exchange. In his thesis “Investigation of Atmospheric Re-Entry for the Space Maneuver Vehicle,” McNabb describes that for a given re-entry trajectory, a TAV

³⁹ J. A. Storch, *Aerodynamic Disturbances on Spacecraft in Free-Molecular Flow* (El Segundo, CA: The Aerospace Corporation, 2002), 3.

⁴⁰ Ibid.

⁴¹ Desmond King-Hele, *Satellite Orbits in an Atmosphere: Theory and Applications* (Glasgow, Scotland: Blackie and Son Ltd., 1987), 23.

⁴² Ibid., 24.

will operate in the rarefied (free molecular), transition (slip-flow), and continuum flow regimes of the upper atmosphere. Defined by the Knudsen number (Kn), or the ratio of the particle mean free path to characteristic length of the TAV aerodynamic chord, McNabb identified rarefied flow as $Kn > 10$, transitional flow as $0.01 \leq Kn \leq 10$, and continuum flow as $Kn < 0.01$.⁴³ As the depth of atmospheric penetration increases during the execution of an aeroassisted maneuver, the atmospheric density increases and, as a result, the flow regime transitions from rarefied to continuum flow as altitude decreases.

With the flow characteristics established for flight in the upper atmosphere, the aerodynamics of a TAV can be determined by either assuming or directly calculating values for the drag and lift coefficients. Consulting a Douglas Aircraft Company technical report entitled “Surface-Particle-Interaction Measurements using Paddlewheel Satellites,” Guettler in his thesis “Satellite Attitude Control using Atmospheric Drag” assumes a constant value drag coefficient of 2.2 for his analysis regarding the employment of aerodynamic torques produced by deployable drag panels as a primary source of satellite attitude control.⁴⁴ A drag coefficient of 2.2 is also given by Vallado, who states that such a value is derived by modeling a satellite operating within the upper atmosphere as a flat plate.⁴⁵ Although greater in magnitude than the value utilized by Guettler, Hall assumed in his thesis “Orbit Maneuver for Responsive Coverage Using Electric Propulsion” a drag coefficient of 3.0, which was posited as one of many “commonly achievable design parameters based upon existing satellite designs.”⁴⁶

⁴³ Dennis J. McNabb, “Investigation of Atmospheric Re-Entry for the Space Maneuver Vehicle” (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2004): 14-15.

⁴⁴ David B. Guettler, “Satellite Attitude Control Using Atmospheric Drag” (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2007): 24.

⁴⁵ Vallado, 549.

⁴⁶ Timothy S. Hall, “Orbit Maneuver for Responsive Coverage Using Electric Propulsion” (MS Thesis, School of Engineering and Management, Air Force Institute of Technology (AU), 2010): 18.

As for direct calculation, Nicholson computes values for the aerodynamic coefficients as a function of angle-of-attack based on empirically-derived equations developed from linearly-interpolated wind tunnel data from tests performed on the Entry Research Vehicle (ERV) within the supersonic velocity range up to Mach 10. Debuted in the conference paper “Performance Evaluation of an Entry Research Vehicle” by Powell, Naftel, and Cunningham, the ERV was a lifting entry test platform designed to investigate maneuvers involving “long downrange, wide cross-range, and synergistic plane changes.”⁴⁷ Similarly, Parish in his thesis “Optimality of Aeroassisted Orbital Plane Changes” also computes values for the aerodynamic coefficients from interpolated transonic and supersonic wind tunnel data, but for the Maneuverable Re-Entry Research Vehicle (MRRV) rather than the ERV. Over the angle-of-attack range of 0 deg to 40 deg, the drag coefficient varies from 0.1 to approximately 1.2 for the ERV, while it varies from 0.03 to approximately 0.6 for the MRRV.⁴⁸ Overall, the preceding values for the vehicle drag coefficient as depicted by Nicholson and Parish are consistent with the research of Rao, Scherich, Cox, and Mosher who, in their conference paper “A Concept for Operationally Responsive Space Mission Planning Using Aeroassisted Orbital Transfer,” utilized a drag coefficient of approximately 0.49 in their study of an aerodynamically maneuverable TAV.⁴⁹

The Atmospheric Flow Environment and Heat Flux

The maturation of ballistic missile technology during the mid-1950s precipitated the need to not only characterize and model re-entry heat flux, but also devise methods by which to mitigate heating effects and forestall mission failure during re-entry. With heat flux analysis on

⁴⁷ Nicholson, 34-35, 144.

⁴⁸ Ibid., 36; Michael S. Parish II, “Optimality of Aeroassisted Orbital Plane Changes” (MS Thesis, Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, 1995): 11-12).

⁴⁹ Anil V. Rao, Arthur E. Scherich, Skylar Cox, and Todd E. Mosher, “A Concept for Operationally Responsive Space Mission Planning Using Aeroassisted Orbital Transfer” (paper presented at the *6th Responsive Space Conference*, Los Angeles, CA, 28 April – 1 May 2008): 3-5.

slender-body ballistic warheads giving way to blunt-body capsules and proposed lifting entry vehicles for manned spaceflight, several experimental techniques were developed to estimate heat flux within the hypersonic flow environment of re-entry. Derived from measuring of heat transfer rates in shock tubes under simulated hypersonic conditions, Detra, Kemp, and Riddell in “Addendum to ‘Heat Transfer to Satellite Vehicles Re-Entering the Atmosphere’” presented a revised empirical equation for stagnation heat flux on a blunt body:⁵⁰

$$\dot{Q}_s = \frac{17,600}{\sqrt{R_N}} \left(\frac{\rho}{\rho_{SL}} \right)^{0.5} \left(\frac{V}{V_{SL}} \right)^{3.15} \left(\frac{h_s - h_w}{h_s - (h_w)_{300 \text{ K}}} \right) \text{ BTU}/(\text{ft}^2 \cdot \text{s}) \quad (2.1)$$

where h_s is the stagnation point enthalpy, h_w is the wall enthalpy, $(h_w)_{300 \text{ K}}$ is the wall enthalpy evaluated at 300 K, R_N is the vehicle nose radius, and $V_{SL} = 26,000 \text{ ft/s}$, a pre-defined sea-level orbital velocity. Identified as being “nearer the mean of the data” than a previous model iteration derived by the same authors, the revised equation “agrees with calculated heat transfer results” for altitude and velocity ranges of $0 \leq h \leq 250,000 \text{ ft}$ ($0 \leq h \leq 76.2 \text{ km}$) and $7,000 \leq V \leq 25,000 \text{ ft/s}$ ($2.1 \leq V \leq 7.6 \text{ km/s}$), respectively, with an accuracy of $\pm 10\%$.⁵¹

Employing a similar empirical form as the Detra *et al.* model, Havey in his 1982 paper “Entry Vehicle Performance in Low-Heat-Load-Trajectories” utilized an equation for stagnation heat flux which accounted for the reduction in heating rate due to a non-zero wall temperature on the vehicle surface:⁵²

$$\dot{Q}_s = 17,700 \left(\frac{\rho}{R_N} \right)^{0.5} \left(\frac{V}{10^4} \right)^{3.07} \left(1 - \frac{h_w}{h_i} \right) \text{ BTU}/(\text{ft}^2 \cdot \text{s}) \quad (2.2)$$

where

⁵⁰ R. W. Detra, N.H. Kemp, and F. R. Riddell, “Addendum to ‘Heat Transfer to Satellite Vehicles Re-Entering the Atmosphere,’” *Jet Propulsion* 27 (1957): 1256.

⁵¹ Ibid., 1257.

⁵² Keith A. Havey Jr., “Entry Vehicle Performance in Low-Heat-Load Trajectories,” *Journal of Spacecraft and Rockets*, 19 (1982): 507.

$$h_w = 0.24T_w \quad h_i = 0.24T_\infty + \frac{V^2}{50,063}$$

The wall temperature, T_w , is determined via the Stefan-Boltzmann Law:

$$\dot{Q} = \varepsilon K_{SB}(T_w^4 - T_\infty^4) \quad (2.3)$$

where $K_{SB} = 0.476 \times 10^{-12}$ BTU/(s · ft² · R⁴) and ε is the emissivity.

For their research in the early 2000s, Rao and several co-authors used a condensed form of the Detra *et al.* model with varying coefficients and units of measure. Removing the enthalpy-differencing term, Rao, Tang, and Hallman utilized the following in their analysis comprising the 2002 paper “Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer”:⁵³

$$\dot{Q}_s = 17,600 \left(\frac{\rho}{\rho_{SL}} \right)^{0.5} \left(\frac{V}{V_{SL}} \right)^{3.15} \text{ BTU/(ft}^2 \cdot \text{s)} = 199,830 \left(\frac{\rho}{\rho_{SL}} \right)^{0.5} \left(\frac{V}{V_{SL}} \right)^{3.15} \text{ kW/m}^2 \quad (2.4)$$

where V_{SL} a function of spherical planetary radius given by $V_{SL} = \sqrt{\mu/r_\oplus}$, rather than a pre-defined value for the sea-level orbital velocity as with the Detra *et al.* model. Maintaining the same equation structure in the 2008 paper “A Concept for Operationally Responsive Space Mission Planning Using Aeroassisted Orbital Transfer,” Rao *et al.* modified the stagnation heat flux coefficient to be $11.357 \text{ kW/m}^2 \approx 1 \text{ BTU/(ft}^2 \cdot \text{s)}$.⁵⁴ Similarly, Darby and Rao in the 2010 paper “Optimal Impulsive LEO to LEO Multiple-Pass Aeroassisted Orbital Transfer for Small Spacecraft” again altered the equation coefficient. Not as drastic as the 2008 paper, the final modification resulted in a 0.02% increase from $199,830 \text{ kW/m}^2$ to $199,870 \text{ kW/m}^2$.⁵⁵

⁵³ Rao et al., “Numerical Optimization Study,” 219.

⁵⁴ Rao et al., “A Concept for Operationally Responsive Space,” 4-5.

⁵⁵ Darby and Rao, “Optimal Impulsive,” 41-42.

Consulting a 1958 General Electric internal study conducted by Brunner and Gallagher entitled “Analysis of the Aerodynamic Heating of a Blunt Hypersonic Glide Vehicle,” Galman presents stagnation heat flux models with zero wall temperature for three-dimensional laminar flow around a sphere,

$$\dot{Q}_s = \frac{3.18}{\sqrt{R_N}} (\rho^{0.5} V^{3.2} \cdot 10^{-9}) \quad (2.5)$$

as well as two-dimensional laminar flow normal to an infinitely-long cylinder:⁵⁶

$$\dot{Q}_s = \frac{3.18}{\sqrt{2R_N}} (\rho^{0.5} V^{3.2} \cdot 10^{-9}) \quad (2.6)$$

In his 1961 paper “Some Fundamental Considerations for Lifting Vehicles in Return from Satellite Orbit,” Galman indicates that “good design practice” for lifting, winged-entry vehicles is to use a large planform nose radius so as to “approach the more favorable” two-dimensional flow model.⁵⁷

Apart from an increase in convective and, specifically, stagnation heat flux, a vehicle re-entering a planetary atmosphere also encounters a likewise increase in radiative heat flux. For Allen and Eggers in their 1958 paper “A Study of the Motion and Aerodynamic Heating of Ballistic Missiles Entering the Earth’s Atmosphere at High Supersonic Speeds,” however, the convective mode of heat flux was deemed to be the dominant form of energy transfer and all radiative heat flux assumed to be negligible.⁵⁸ Qualifying Allen and Eggers’ assertion, Moore in his contribution to Loh’s 1968 work *Entry and Planetary Entry Physics and Technology*:

⁵⁶ Barry A. Galman, “Some Fundamental Considerations for Lifting Vehicles in Return from Satellite Orbit,” *Planetary and Space Science*, 4 (1961): 400.

⁵⁷ Ibid.

⁵⁸ H. J. Allen and A. J. Eggers, Jr., “A Study of the Motion and Aerodynamic Heating of Ballistic Missiles Entering the Earth’s Atmosphere at High Supersonic Speeds,” *NACA TR 1381* (Moffett Field, CA: AMES Aeronautical Laboratory, 1958), 1129.

Dynamics, Physics, Radiation, Heat Transfer, and Ablation states that radiative heat flux is a “particularly sensitive function of flight velocity” and it “takes off” at speeds just beyond orbital.”⁵⁹ As examples, Moore compares the stagnation and radiative heat flux of an intercontinental ballistic missile (ICBM) entering the Earth’s atmosphere with that of a probe entering the Martian atmosphere. With the former example, an ICBM velocity of approximately 20,000 ft/s (6.1 km/s) produces a stagnation heat flux 2.5-3.0 times greater than the radiative heat flux; for the latter example, the probe entry velocity of 40,000 ft/s (12.2 km/s) yields a radiative heat flux 10 times greater than the stagnation heat flux.⁶⁰

Providing a more precise threshold for radiative heat flux dominance, Olfe in the 1968 book *Radiation and Re-Entry* states that as the re-entry velocity increases towards that of a parabolic orbit (~11.19 km/s), the radiative heat flux “rapidly overtakes the aerodynamic heat transfer” and can “appreciably affect the flow field.”⁶¹ Olfe also conveys that as the re-entry velocity increase above the parabolic value, the radiative energy loss from the shock layer “approaches the magnitude of the flow energy.”⁶² Although published earlier in 1961, Eggers’ and Wong’s paper “Motion and Heating of Lifting Vehicles during Atmosphere Entry” affirms Olfe’s threshold and posits that the maximum radiative heat flux corresponds to a velocity of approximately 36,000 ft/s (11.0 km/s).⁶³

⁵⁹ F. K. Moore, “Entry Radiative Transfer,” in *Re-Entry and Planetary Entry Physics and Technology: Dynamics, Physics, Radiation, Heat Transfer, and Ablation*, ed. W. H. T. Loh (New York, NY: Springer-Verlag New York Inc., 1968), 343.

⁶⁰ Ibid.

⁶¹ Daniel B. Olfe, “Radiation Gasdynamics,” in *Radiation and Re-Entry*, ed. S. S. Penner and Daniel B. Olfe (New York, NY: Academic Press Inc., 1968), 271.

⁶² Ibid., 272.

⁶³ A. J. Eggers Jr. and Thomas J. Wong, “Motion and Heating of Lifting Vehicles during Atmosphere Entry,” *American Rocket Society (ARS) Journal*, 31 (1961): 1370.

Summary

Upon review of the relevant research pertaining to aeroassisted maneuvers, it can be asserted that despite complexities due to high temperature and varying density gas dynamics, the upper atmosphere provides an advantageous environment within which maneuvers can be executed to alter a TAV's orbital states, such as inclination and semi-major axis. Whether performed by small vehicles with an initial mass less than 1000 kg, or larger vehicles with an initial mass greater than 5000 kg, preceding research indicates that aeroassisted maneuvers generally require less ΔV than a purely propulsive maneuver conducted in the vacuum environment to produce desired changes in orbital states and geometry. While the prospect of responsive spacecraft and global reach has been demonstrated by a satellite with electric propulsion performing exo-atmospheric maneuvers, the current literature is limited regarding the reachability performance potential of aeroassisted maneuvers outside the realm of single- and multi-objective comparative optimization problems. As a result, the present research serves to augment the current literature through an application-based analysis of aeroassisted maneuver performance for the intent of achieving not only terrestrial, but also LEO reachability for a TAV initiating from the LEO altitude regime.

III. Methodology

Chapter Overview

The purpose of this chapter is to provide an overview of the assumptions, limitations, and algorithms underpinning the trajectory dynamics model, a simulation tool capable of modeling both exo- and trans-atmospheric maneuvers. In addition to the verification of the trajectory model by duplicating the Apollo 10 re-entry trajectory, models associated with simulating the atmospheric density, gravitational potential, and re-entry heat flux is be discussed.

Assumptions and Limitations

Planetary Ellipticity

Unlike the planet Venus which features a nearly spherical shape, Earth is a rotationally symmetric ellipsoid that revolves about its minor axis. Also known as an oblate spheroid, the Earth's shape is characterized by a flattening at the poles, thus creating a polar (minor) axis shorter in diameter than the equatorial (major) axis.⁶⁴ Depicted in the following figure, the ellipticity of the Earth presents two different means of expressing radial position: (1) *Geocentric* latitude, ϕ , which is measured with respect to the planetary center-of-mass; and (2) *geodetic* latitude, ϕ_{gd} , which is offset from the planetary center-of-mass and measured with respect to the TAV such that the position vector is perpendicular to a plane tangent to the planetary surface.

⁶⁴ Vallado, 142.

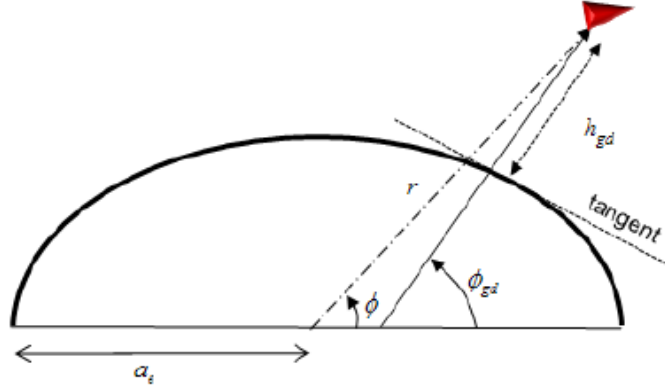


Figure 3.1. Comparison of Geocentric and Geodetic Latitude⁶⁵

Since the equations of motion outlined later in this chapter are formulated in terms of the geocentric representation, any simulation initial conditions featuring geodetic altitude and latitude must be converted into geocentric values by employing analytical expressions obtained from Long's paper "General-Altitude Transformations between Geocentric and Geodetic Coordinates." Formulated as truncated series expansions in powers of the Earth's flattening, f , the following second-order equations are functions of geodetic altitude and latitude:⁶⁶

$$r = (h_{gd} + 1) + \left[-\frac{1}{2}(1 - \cos 2\phi_{gd}) \right] f + \left\{ \left[\frac{1}{4(h_{gd} + 1)} + \frac{1}{16} \right] (1 - \cos 4\phi_{gd}) \right\} f^2 \quad (3.1)$$

$$\begin{aligned} \phi = \phi_{gd} + & \left(\frac{-\sin 2\phi_{gd}}{h_{gd} + 1} \right) f \\ & + \left\{ \frac{-\sin 2\phi_{gd}}{2(h_{gd} + 1)^2} + \left[\frac{1}{4(h_{gd} + 1)^2} - \frac{1}{4(h_{gd} + 1)} \right] \sin 4\phi_{gd} \right\} f^2 \end{aligned} \quad (3.2)$$

⁶⁵ Hicks, 382.

⁶⁶ S.A.T. Long, "General-Altitude Transformations between Geocentric and Geodetic Coordinates," *Celestial Mechanics* 12 (1975): 228.

When Eqs. (3.1) and (3.2) are evaluated with a geodetic altitude of $h_{gd} = 0.0$ km for sea-level and a geodetic latitude range of $-90 \text{ deg} \leq \phi_{gd} \leq 90 \text{ deg}$, the corresponding radial distance from the planetary center of mass to sea-level for a given latitudinal position on the oblate spheroid can be determined. Illustrated in Fig. 3.2, the oblate spheroid model creates a radial difference of 21.385 km at the poles and 0.0 km at the equator when compared with an equivalent spherical model. Due to this disparity in radial distance, the Earth planetary model retains the characteristics of an oblate spheroid with an ellipticity of $\epsilon = 0.08181919$.⁶⁷

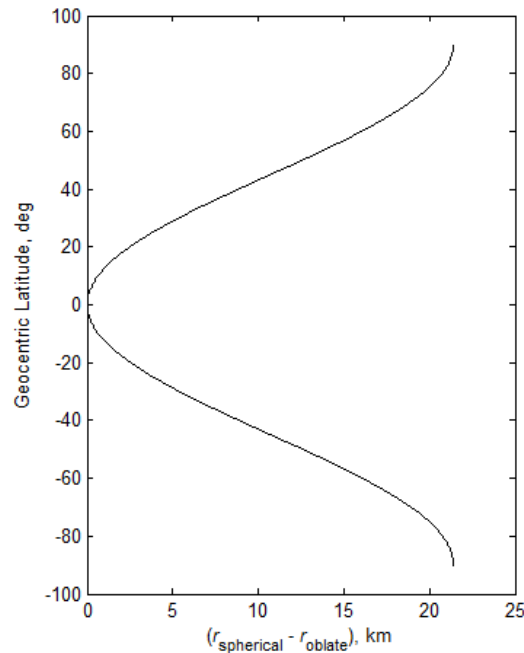


Figure 3.2. Radial Distance Deviation between Spherical and Oblate Spheroid Models

Atmospheric Density and Dynamics

For most spacecraft, the nominal operating altitude is located above the upper limit of the sensible atmosphere of approximately 120 km. Based on this demarcation, orbital perturbations arising from atmospheric drag are only a concern for spacecraft in the lower reaches of LEO

⁶⁷ Hicks, 382.

below an altitude of 400 km. Even for these spacecraft, however, interaction with the rarefied flow environment of the exosphere is contingent on the solar cycle and the expansion of the atmosphere due to increased solar and resultant geomagnetic activity. Conversely, spacecraft categorized as TAVs possess the ability to perform aeroassisted maneuvers and exploit atmospheric drag to alter orbital elements such as inclination or right ascension of the ascending node.

With aeroassisted, trans-atmospheric trajectories producing a perigee of less than 120 km, a model of atmospheric density is required to simulate the spacecraft's aerodynamic characteristics, specifically the drag and lift force generated at a particular altitude. The simplest model assumes that atmospheric density decreases exponentially with increasing altitude:⁶⁸

$$\rho(r) = \rho_{SL} e^{-\beta(r-r_{\oplus})} \quad (3.3)$$

where the scale height, β , is constant throughout the atmosphere. Formulated in terms of a spherical planetary model, Eq. (3.3) determines the atmospheric density at a specific altitude defined by a non-varying radius r_{\oplus} from the planetary center of mass to the surface. For a spherical planet, the altitude $h = r - r_{\oplus}$ is both geocentric and geodetic in nature, and is measured along an imaginary vertical line perpendicular to the planetary surface and passing through the TAV center-of-mass.⁶⁹ When the planetary model is changed from spherical to oblate spheroid, Eq. (3.3) then requires a geodetic altitude at which to calculate the atmospheric density. In order to reflect this subtlety, the following represents the modified exponential density model for an oblate spheroid model:

⁶⁸ Ibid., 68.

⁶⁹ Vladimir A. Chobotov, *Orbital Mechanics*, Third Edition (Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2002), 72.

$$\rho(h_{gd}) = \rho_{SL} e^{-\beta(h_{gd})} \quad (3.4)$$

Since geocentric radius represents a specified state within the equations of motion for atmospheric re-entry rather than geodetic altitude, a conversion must be performed to derive the geodetic altitude value in order to calculate the atmospheric density for a given geocentric radius.

Rather than calculating the geodetic altitude and associated geodetic latitude simultaneously via an iterative algorithm as described in Hicks' text *Introduction to Astrodynamic Re-Entry*, analytical expressions can be implemented *a posteriori* from Long's aforementioned paper. Also formulated as truncated series expansions in powers of the Earth's flattening, f , the following second-order equations are functions of geocentric coordinates in units of the Earth's equatorial radius:⁷⁰

$$h_{gd} = (r - 1) + \left[\frac{1}{2} (1 - \cos 2\phi) \right] f + \left[\left(\frac{1}{4r} - \frac{1}{16} \right) (1 - \cos 4\phi) \right] f^2 \quad (3.5)$$

$$\phi_{gd} = \phi + \left(\frac{\sin 2\phi}{r} \right) f + \left[\left(\frac{1}{r^2} - \frac{1}{4r} \right) \sin 4\phi \right] f^2 \quad (3.6)$$

As an alternative to the exponential density model, Vinh, Busemann, and Culp in their book *Hypersonic and Planetary Entry Flight Mechanics* provide an equation which accounts for variation in both scale height and molecular scale temperature throughout the atmosphere. Expressed in terms of geodetic altitude, the dual variation model is:⁷¹

$$\rho(h_{gd}) = \rho_i \left[\left(1 + \delta_{TM} \left(\frac{h_{gd} - h_i}{r_{\oplus}} \right) \right)^{-1} \right] \cdot \left[\left(1 + \delta_H \left(\frac{h_{gd} - h_i}{r_{\oplus}} \right) \right)^{-1} \right]^{\frac{1}{\alpha}} \quad (3.7)$$

⁷⁰ Long, 225-226, 228.

⁷¹ Nguyen X. Vinh, Adolf Busemann, and Robert D. Culp, *Hypersonic and Planetary Entry Flight Mechanics* (Ann Arbor, MI: The University of Michigan Press, 1980), 9.

where the subscript i represents an index for the division of the atmosphere into seven sections between $54 \leq h_{gd} \leq 300$ km, and δ_H , δ_{TM} are dimensionless parameters related to scale height and molecular scale temperature, respectively, for the seven altitude-demarcated sections of the atmosphere. In their discussion, Vinh *et al.* identify that the dual variation model can be simplified by noting that δ_H , δ_{TM} are approximately equal throughout the seven sections and thus reduce Eq. (3.7) into a *single variation* model, which only accounts for variation in scale height:⁷²

$$\rho(h_{gd}) = \rho_i \left[\left(1 + \delta_H \left(\frac{h_{gd} - h_i}{r_\oplus} \right) \right)^{-1} \right]^{\frac{1+\alpha}{\alpha}} \quad (3.8)$$

So as to evaluate the relative capability of the exponential, single, and dual variation models to accurately estimate atmospheric density, the solutions of each were compared with density results from the MSIS-E-90 density model within the altitude range $0 \leq h_{gd} \leq 1000$ km for the sample dates 01 January 2000-2012. With the first iteration being developed in the late 1970s at the NASA/Goddard Space Flight Center, the mass spectrometer-incoherent scatter (MSIS) series of atmospheric density models are empirical in nature and assimilate in situ mass spectrometer measurements of temperature and composition, as well as “temperatures inferred from incoherent scatter radar data.”⁷³ Although other high fidelity atmospheric models exist, Akins, Healy, Coffey, and Picone in their paper “Comparison of MSIS and Jacchia Atmospheric Density Models for Orbit Determination and Propagation” indicate that the atmospheric physics community has “validated the [MSIS] model” via direct measurement of density and has

⁷² Ibid.

⁷³ National Research Council, Aeronautics and Space Engineering Board, *Continuing Kepler’s Quest: Assessing Air Force Space Command’s Astrodynamics Standards* (Washington, D.C.: The National Academies Press, 2012), 23.

demonstrated the superiority of the MSIS series over older models such as Jacchia-70.⁷⁴ As a result, the MSIS-derived density solutions are deemed admissible as “truth” data for comparative and root-mean square (RMS) error analysis.

In addition to the MSIS-E-90 data, an atmospheric density profile was obtained from the AGI analysis module *Astrogator* within Systems Toolkit[®] (STK) and plotted against the exponential and single variation density models. By default, *Astrogator* employs the US 1976 Standard Atmospheric Density Model and only provides density estimates along the trajectory rather than a specified altitude regime. Due to the resolution of the following figures, the dual variation curve is omitted since it nearly coincides with the single variation solution and any differences between the two models are not readily discernible. Also, only the MSIS data for 01 January 2012 is plotted in order to provide a single example of the thirteen data sets obtained from the MSIS model. All thirteen sets as well as the STK[®] density data are illustrated in Fig. 3.4. As seen in the Fig. 3.3, the exponential, single, and dual variation models are insufficient in modeling the MSIS and STK[®] data over the entire altitude range $0 \leq h_{gd} \leq 1000$ km. For the exponential model, the accuracy of density predictions is superior to the single and dual variations models and features the least deviation with the MSIS and STK[®] data until an altitude of approximately 84 km where solution divergence initiates. More limited, the single and dual variation models are only applicable within the range $54 \leq h_{gd} \leq 300$ km and are thus unable to provide density predictions for 75.4% of the altitude range $0 \leq h_{gd} \leq 1000$ km.

⁷⁴ Keith Akins, Liam Healy, Shannon Coffey, and Mike Picone, “Comparison of MSIS and Jacchia Atmospheric Density Models for Orbit Determination and Propagation” (paper presented at the *13th AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico, 9-13 February 2003), 3.

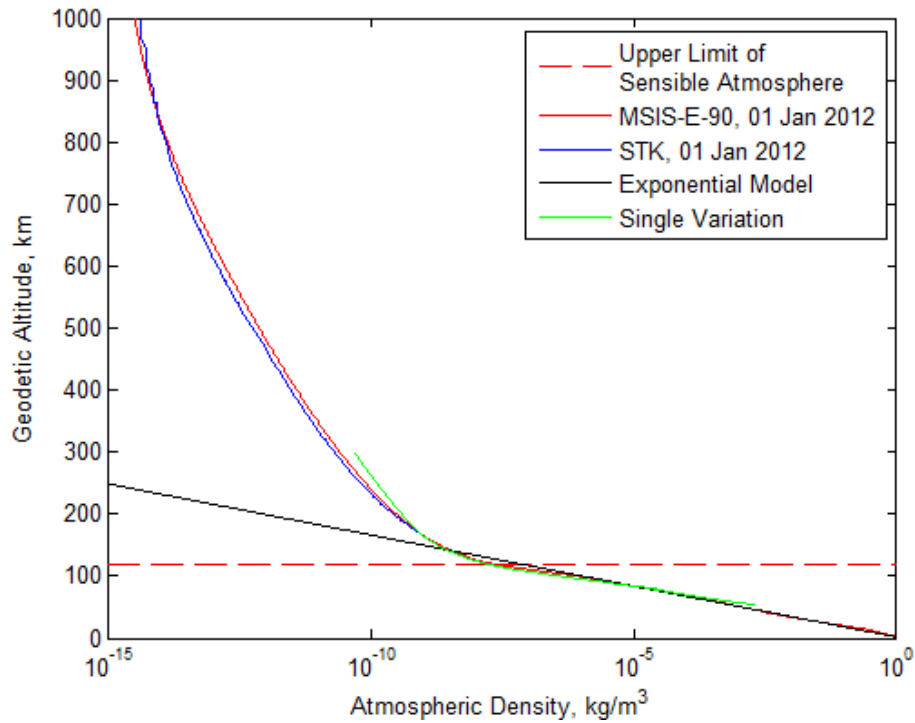


Figure 3.3. Initial Comparison of Atmospheric Density Models with MSIS-E-90 and STK[®] Density Data for 01 January 2012

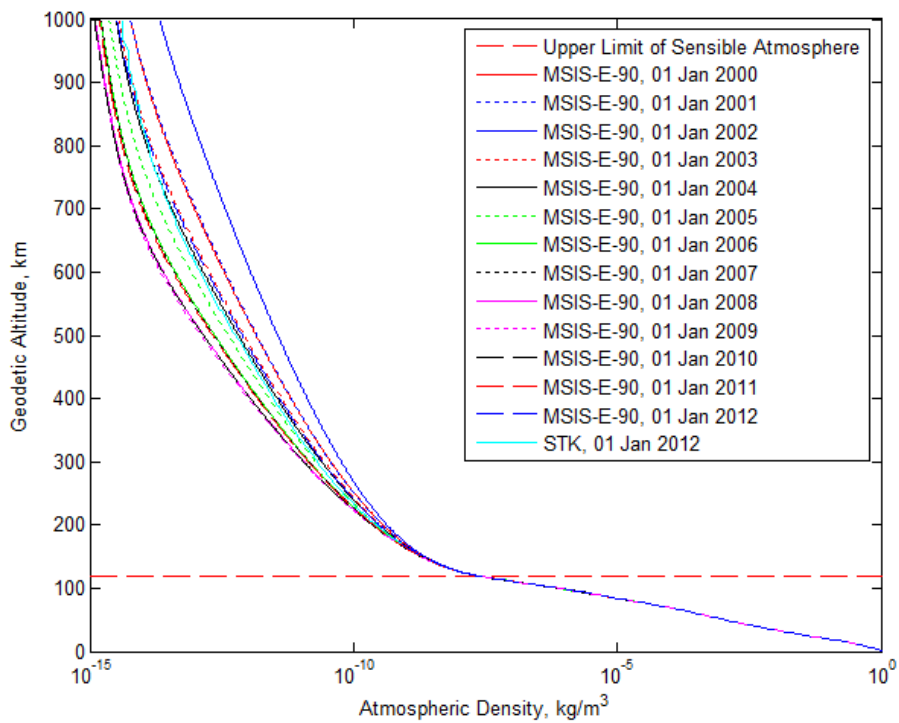


Figure 3.4. Comparison of MSIS-E-90 and STK[®] Density Data

Faced with the solution inadequacies of the exponential, single, and dual variation models as individual equations, a piecewise-continuous function – or *Combined Model* – was developed, and is:

$$\rho(h_{gd}) = \begin{cases} \rho_{SL} e^{-\beta(h_{gd})} & , h_{gd} < 84 \text{ km} \\ \rho_i \left[\left(1 + \delta_H \left(\frac{h_{gd} - h_i}{r_{\oplus}} \right) \right)^{-1} \right]^{\frac{1+\alpha}{\alpha}} & , 84 \leq h_{gd} \leq 120 \text{ km} \\ (4.50847623 \times 10^7) \cdot (h_{gd})^{-7.44605852} & , 120 < h_{gd} \leq 1000 \text{ km} \end{cases} \quad (3.9)$$

For all altitudes above the 1000 km threshold, the density is assumed to be 0.0 kg/m³.

Parameters given in the single variation segment of Eq. (3.9) are listed in the following table:

Table 3.1. Atmospheric Density Model Parameters

<i>Altitude Section</i>	h_i, km	$\rho_i, \text{kg/m}^3$	α	δ_H
$84 \leq h_{gd} \leq 90 \text{ km}$	85	7.726×10^{-6}	0.1545455	197.9740
$91 \leq h_{gd} \leq 106 \text{ km}$	99	4.504×10^{-7}	0.1189286	128.4577
$107 \leq h_{gd} \leq 120 \text{ km}$	110	5.930×10^{-8}	0.5925240	432.8484

While the first two equations represent the exponential and single variation models, the third is a power model formulated through regression analysis of the MSIS and STK[®] data. Since atmospheric density changes with not only date and local time, but also geographical location, both the MSIS and STK[®] data sets were obtained for the date 01 January at 12:00:00.00 Universal Time for the latitude/longitude coordinates $(\theta, \phi) = (0,0)$ deg. Unlike the MSIS data which is defined for the year range 2000-2012, the STK[®] data only represents the year 2012 due to a preliminary RMS error analysis conducted with the following expression:⁷⁵

$$RMS_X = \sqrt{\frac{\sum_{i=1}^n \left[(X(t_i))_{simulation} - (X(t_i))_{truth} \right]^2}{n}} \quad (3.10)$$

⁷⁵ Hicks, 394.

with the “simulation” data representing the years 2000-2011 and the “truth” data the year 2012. From this analysis, a deviation of approximately $2 \times 10^{-16} \text{ kg/m}^3$ was calculated between the data for 2012 and the years 2000-2011, thus enabling the data for the years 2000-2011 to be excluded from all subsequent comparative analysis.

As shown in Fig. 3.5, the Combined Model maintains the least deviation with MSIS and STK[®] data for the altitude range $0 \leq h_{gd} \leq 1000 \text{ km}$. Quantified in terms of RMS error, the Combined Model deviates from the MSIS data by approximately $1.2 \times 10^{-2} \text{ kg/m}^3$ and the STK[®] data by $9.181 \times 10^{-11} \text{ kg/m}^3$. Based its ability to predict atmospheric density from the troposphere through to upper reaches of the xenosphere, the Combined Model is implemented as the density model for all aeroassisted maneuver analysis.

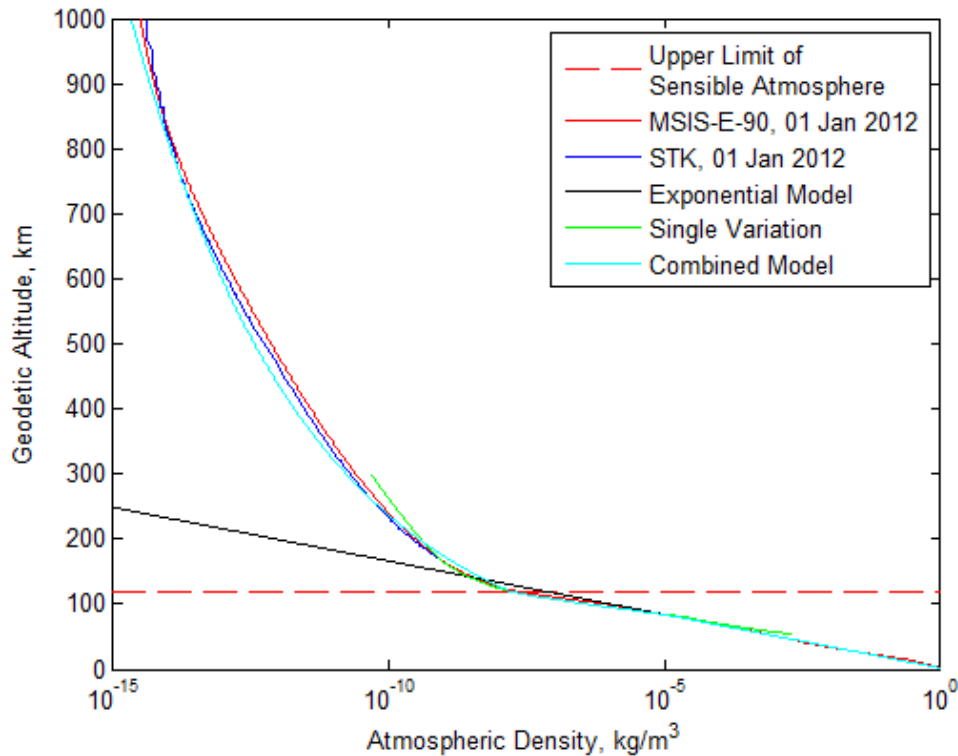


Figure 3.5. Comparison of Combined Atmospheric Density Model with MSIS-E-90 and STK[®] Density Data for 01 January 2012

Table 3.2. RMS Error for Combined Density Model Compared with MSIS-E-90 and STK[®] Density Data

<i>Data Set</i>	<i>n</i>	<i>RMS Error, kg/m³</i>
MSIS-E-90, 01 Jan 2000	1000	1.228×10^{-2}
MSIS-E-90, 01 Jan 2001	1000	1.232×10^{-2}
MSIS-E-90, 01 Jan 2002	1000	1.231×10^{-2}
MSIS-E-90, 01 Jan 2003	1000	1.230×10^{-2}
MSIS-E-90, 01 Jan 2004	1000	1.227×10^{-2}
MSIS-E-90, 01 Jan 2005	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2006	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2007	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2008	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2009	1000	1.225×10^{-2}
MSIS-E-90, 01 Jan 2010	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2011	1000	1.226×10^{-2}
MSIS-E-90, 01 Jan 2012	1000	1.230×10^{-2}
STK [®] , 01 Jan 2012	570	9.181×10^{-11}

Besides variations in density with altitude, the atmosphere is also highly dynamic and rotates, albeit with a lower angular velocity, concomitant to the planet. Vinh *et al.* state that the maximum rotational velocity of the atmosphere at the equator is approximately six percent of the circular orbit velocity at low altitude. Furthermore, the aerodynamic force due to atmospheric rotation has a maximum of about 12% of the aerodynamic force arising due to the vehicle's velocity. Although dependent on not only vehicle velocity, but also latitude, and the inclination of the trajectory to the equator, Vinh *et al.* conclude that the effects of atmospheric rotation are “so slight” and that any errors introduced by estimating an entry vehicle's drag and lift coefficients exceeds the error caused by neglecting atmospheric rotation.⁷⁶ Due to the complexities of and inherent error associated with endeavoring to model independent rotation, the atmosphere is assumed to be rotating at the same angular velocity as the planetary model.

⁷⁶ Vinh et al., *Hypersonic and Planetary Entry Flight Mechanics*, 3.

TAV Mass Properties

As with any object, the mass of a TAV is distributed throughout the envelope of the vehicle's three-dimensional shape, with such a distribution expressed as a mass moment of inertia calculated about the principal axes of the vehicle's body-fixed coordinate frame. Although a more accurate representation of the vehicle mass, the calculation of mass moment of inertia values is contingent on the implicit assumption that the vehicle is a rigid body and, therefore, does not deform nor change shape.⁷⁷ As a simplifying alternative, the TAV is modeled as a point mass with the total force, \vec{F} , acting on the point mass at any instant in time expressed by the following:⁷⁸

$$\vec{F} = \vec{T} + \vec{A} + m\vec{g} \quad (3.10)$$

where \vec{T} is the thrust force, \vec{A} is the aerodynamic force comprised of drag and lift components, and \vec{g} is the gravitational force.

In addition to the point mass simplification, the TAV is assumed to maintain a constant mass, with propellant only being expended prior to and/or following a maneuver. Due to the high-temperature molecular interactions between the vehicle surface and the various gaseous species of the “chemically reacting boundary layer” during an aeroassisted maneuver, the constant mass simplification is also maintained within the hypersonic re-entry flow environment by assuming the employment of a non-ablative thermal control subsystem on the vehicle surface.⁷⁹

⁷⁷ Anthony Bedford and Wallace Fowler, *Engineering Mechanics: Dynamics*, Fourth Edition (Upper Saddle River, NJ: Pearson Prentice Hall, 2005), 280, 398.

⁷⁸ Hicks, 37.

⁷⁹ Anderson, 17.

Total Force Properties

Defined in terms of the vehicle-pointing reference frame ($OX_2Y_2Z_2$), with the origin coincident with the point mass, the gravitational force acts along the radial position vector originating from the planetary center of mass and is aligned with the x_2 -axis. Not aligned with any specific axis within the vehicle-pointing system, however, the aerodynamic force can be described in relation to the TAV's velocity vector, with the lift and drag forces acting in directions perpendicular to and opposite the velocity vector, respectively. While the gravitational force is expressed in the vehicle-pointing system, both the aerodynamic and thrust forces can be described by a coordinate reference system fixed to the TAV center of mass.⁸⁰ The relationship of the thrust force to a sample TAV's aerodynamic lift, drag, and velocity vector is shown in the following depiction of the North American-Rockwell Space Shuttle concept:

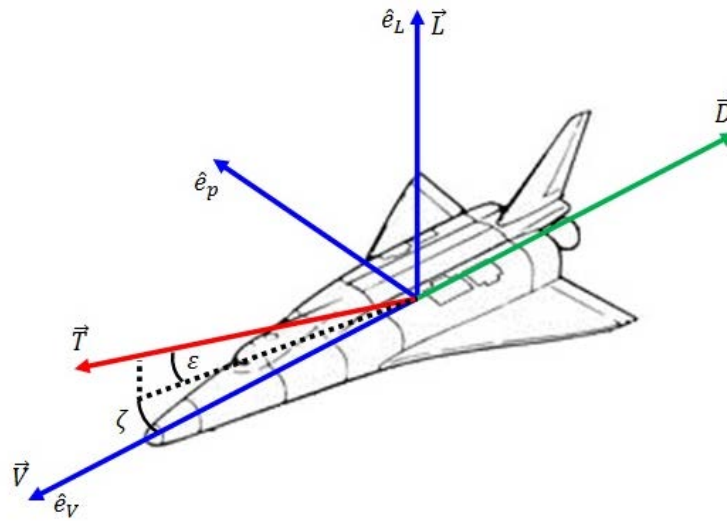


Figure 3.6. Vehicle Reference Frame and Vector Definition for Sample TAV⁸¹

⁸⁰ Hicks, 29, 43-46.

⁸¹ Ibid., 47; T. A. Heppenheimer, *The Space Shuttle Decision: NASA's Search for a Reusable Space Vehicle* (Washington, D.C.: National Aeronautics and Space Administration, 1999), 333.

When examined separately, the aerodynamic and thrust forces provide the impetus for further simplifying assumptions. The aerodynamic force is a dynamic quantity during an aeroassisted maneuver due to the geometry of the TAV relative to the hypersonic re-entry flow environment, the viscous interactions between the rarefied gaseous species of the upper atmosphere and the vehicle surface, and the decrease in air density resulting from increases in temperature. In order to simplify these dynamical flow complexities, the drag and lift coefficients are modeled as constant values. Produced by a notional TAV propulsion subsystem, the thrust force is modeled as impulsive and capable of being applied instantaneously.

Earth-Based Constants

Various planetary and atmospheric parameters are modeled as constant values and are outlined in Table 3.3:⁸²

Table 3.3. Earth-Based Constants

<i>Constant</i>	<i>Value</i>
Gravitational Parameter, μ_{\oplus}	398600.442 km ³ /s ²
Gravitational Acceleration at Sea-Level, g_{SL}	9.798 m/s ²
Planetary Radius, r_{\oplus}	6378.137 km
Atmospheric Scale Height, β	0.14 km ⁻¹
Atmospheric Density at Sea-Level, ρ_{SL}	1.225 kg/m ³

⁸² Hicks, 381; Vallado, 138, 140.

Trajectory Dynamics Model Development

Described by a system of six nonlinear differential equations, re-entry and aeroassisted maneuvers are simulated by the following set of kinematic and dynamical equations:⁸³

$$\dot{r} = {}^R V \sin \gamma \quad (3.11)$$

$$\dot{\theta} = \frac{{}^R V \cos \gamma \cos \psi}{r \cos \phi} \quad (3.12)$$

$$\dot{\phi} = \frac{{}^R V \cos \gamma \sin \psi}{r} \quad (3.13)$$

$${}^R \dot{V} = \frac{T}{m} (\cos \zeta \cos \varepsilon) - \frac{D}{m} - g(r) \sin \gamma + r \omega_{\oplus}^2 \cos \phi (\cos \phi \sin \gamma - \sin \phi \sin \psi \cos \gamma) \quad (3.14)$$

$$\begin{aligned} {}^R V \dot{\gamma} = & \frac{T}{m} (\sin \zeta \sin \sigma + \cos \zeta \sin \varepsilon \cos \sigma) + \frac{L}{m} \cos \sigma - g(r) \cos \gamma + \frac{{}^R V^2}{r} \cos \gamma + \\ & 2 {}^R V \omega_{\oplus} \cos \phi \cos \psi + r \omega_{\oplus}^2 \cos \phi (\cos \phi \cos \gamma - \sin \phi \sin \psi \sin \gamma) \end{aligned} \quad (3.15)$$

$$\begin{aligned} {}^R V \dot{\psi} = & \frac{1}{m \cos \gamma} [T (\cos \zeta \sin \varepsilon \sin \sigma - \sin \zeta \cos \sigma) + L \sin \sigma] - \frac{{}^R V^2}{r} \cos \gamma \cos \psi \tan \phi \\ & + 2 {}^R V \omega_{\oplus} (\sin \psi \cos \phi \tan \gamma - \sin \phi) - \frac{r \omega_{\oplus}^2}{\cos \gamma} \sin \phi \cos \phi \cos \psi \end{aligned} \quad (3.16)$$

where the drag and lift forces are computed, respectively, by:

$$D = \frac{1}{2} \rho C_D S {}^R V^2 \quad L = \frac{1}{2} \rho C_L S {}^R V^2$$

Based on the assumption that the TAV is non-thrusting, the preceding equations of motion can be simplified to the following with the thrust force, T , equal to zero:

$$\dot{r} = {}^R V \sin \gamma \quad (3.17)$$

$$\dot{\theta} = \frac{{}^R V \cos \gamma \cos \psi}{r \cos \phi} \quad (3.18)$$

$$\dot{\phi} = \frac{{}^R V \cos \gamma \sin \psi}{r} \quad (3.19)$$

$${}^R \dot{V} = -\frac{D}{m} - g(r) \sin \gamma + r \omega_{\oplus}^2 \cos \phi (\cos \phi \sin \gamma - \sin \phi \sin \psi \cos \gamma) \quad (3.20)$$

⁸³ Ibid., 42, 52.

$${}^RV\dot{\gamma} = \frac{L}{m} \cos \sigma - g(r) \cos \gamma + \frac{{}^RV^2}{r} \cos \gamma + 2 {}^RV\omega_{\oplus} \cos \phi \cos \psi + r\omega_{\oplus}^2 \cos \phi (\cos \phi \cos \gamma - \sin \phi \sin \psi \sin \gamma) \quad (3.21)$$

$${}^RV\dot{\psi} = \frac{L \sin \sigma}{m \cos \gamma} - \frac{{}^RV^2}{r} \cos \gamma \cos \psi \tan \phi + 2 {}^RV\omega_{\oplus} (\sin \psi \cos \phi \tan \gamma - \sin \phi) - \frac{r\omega_{\oplus}^2}{\cos \gamma} \sin \phi \cos \phi \cos \psi \quad (3.22)$$

In the Hicks formulation of the equations of motion, gravitational acceleration is defined in terms of the TAV radial position from the center of a spherical, axisymmetric planet:

$$g(r) = g_{SL} \left(\frac{r_{\oplus}}{r} \right)^2 = \frac{\mu}{r^2} \quad (3.23)$$

Representing the spherical (Newtonian) gravity model, Eq. (3.23) neglects variations in the Earth's gravitational potential due to a non-uniform mass distribution and planetary ellipticity, or oblateness. If the oblate spheroid assumption is implemented, then the trajectory dynamics model will utilize the higher-order J_2 -gravity model which accounts for gravitational potential variations due to ellipticity. In his book *Atmospheric and Space Flight Dynamics*, Tewari derives vector-component expressions for the acceleration due to gravity of a non-spherical, axisymmetric planet. As the foundation of his formulation, Tewari employs spherical harmonics to model the variations in the Earth's gravitational potential deemed negligible by the spherical gravity model. Components of the following co-latitude (φ) dependent equations, the spherical harmonics are given by the Earth-specific Jeffrey constants $J_2 = 0.00108263$, $J_3 = -0.00000254$, $J_4 = -0.00000161$, while the term $P_n(\cos \varphi)$ represents an n^{th} -order Legendre polynomial:⁸⁴

⁸⁴ Ashish Tewari, *Atmospheric and Space Flight Dynamics* (Boston, MA: Birkhäuser, 2007), 51-52.

$$g_r = \frac{\mu}{r^2} \left[1 - 3J_2 \left(\frac{r_\oplus}{r} \right)^2 P_2(\cos \varphi) - 4J_3 \left(\frac{r_\oplus}{r} \right)^3 P_3(\cos \varphi) - 5J_4 \left(\frac{r_\oplus}{r} \right)^4 P_4(\cos \varphi) \right]$$

$$g_\varphi = \frac{3\mu}{r^2} \left(\frac{r_\oplus}{r} \right)^2 \sin \varphi \cos \varphi \left[J_2 + \frac{1}{2} J_3 \left(\frac{r_\oplus}{r} \right) \sec \varphi (5 \cos^2 \varphi - 1) + \frac{5}{6} J_4 \left(\frac{r_\oplus}{r} \right)^2 (7 \cos^2 \varphi - 1) \right]$$

Expanding the Legendre polynomials and replacing the co-latitude variables with that of geocentric latitude via the co-function trigonometric identity, the preceding equations for the radial and transverse components of gravitational acceleration become:

$$g_r = \frac{\mu}{r^2} \left[1 - 3J_2 \left(\frac{r_\oplus}{r} \right)^2 \cdot \left(\frac{1}{2} (3 \sin^2 \phi - 1) \right) - 4J_3 \left(\frac{r_\oplus}{r} \right)^3 \cdot \left(\frac{1}{2} (5 \sin^3 \phi - 3 \sin \phi) \right) - 5J_4 \left(\frac{r_\oplus}{r} \right)^4 \cdot \left(\frac{1}{8} (35 \sin^4 \phi - 30 \sin^2 \phi + 3) \right) \right] \quad (3.24)$$

$$g_\phi = \frac{3\mu}{r^2} \left(\frac{r_\oplus}{r} \right)^2 \cos \phi \sin \phi \left[J_2 + \frac{1}{2} J_3 \left(\frac{r_\oplus}{r} \right) \csc \phi (5 \sin^2 \phi - 1) + \frac{5}{6} J_4 \left(\frac{r_\oplus}{r} \right)^2 (7 \sin^2 \phi - 1) \right] \quad (3.25)$$

Evaluating Eq. (3.24) at an altitude of 1000 km over the equator gives a radial gravitational acceleration of 7.33114498 m/s². By assuming that the contribution of both the J_3 and J_4 Jeffrey constants are negligible, however, the resulting gravitational acceleration decreases to 7.33113256 m/s², which yields a deviation of 1.2420×10^{-5} m/s² from the original value. Based on the magnitude of this deviation, the negligibility assumption proffered for the J_3 and J_4 Jeffrey constants can be maintained, thus simplifying and transforming the J_4 -gravity model into the J_2 -gravity model which only accounts for planetary oblateness:

$$g_r = \frac{\mu}{r^2} \left[1 - 3J_2 \left(\frac{r_\oplus}{r} \right)^2 \cdot \left(\frac{1}{2} (3 \sin^2 \phi - 1) \right) \right] \quad (3.26)$$

$$g_\phi = \frac{3\mu}{r^2} \cdot J_2 \cdot \left(\frac{r_\oplus}{r} \right)^2 \cos \phi \sin \phi \quad (3.27)$$

A final simplification can be made by assuming that the contribution of the J_2 Jeffrey constant is negligible, thus creating the initial spherical gravity model given by Eq. (3.23).

In order to model J_2 -gravity effects, the equations of motion are modified so that the gravitational acceleration consists of both radial (g_r) and transverse (g_ϕ) components. Since gravitational acceleration only appears as a parameter in the trajectory force equations, then only the modified versions of Eqs. (3.14) - (3.15) are presented:⁸⁵

$${}^R\dot{V} = -\frac{D}{m} - g_r \sin \gamma - g_\phi \sin \gamma \cos \gamma + r\omega_\oplus^2 \cos \phi (\cos \phi \sin \gamma - \sin \phi \sin \psi \cos \gamma) \quad (3.28)$$

$$\begin{aligned} {}^RV\dot{\gamma} = & \frac{L}{m} \cos \sigma - g_r \cos \gamma + g_\phi \sin^2 \gamma + \frac{{}^RV^2}{r} \cos \gamma + 2 {}^RV\omega_\oplus \cos \phi \cos \psi + \\ & r\omega_\oplus^2 \cos \phi (\cos \phi \cos \gamma - \sin \phi \sin \psi \sin \gamma) \end{aligned} \quad (3.29)$$

$$\begin{aligned} {}^RV\dot{\psi} = & \frac{L \sin \sigma}{m \cos \gamma} - g_\phi \frac{\cos \psi}{\cos \gamma} - \frac{{}^RV^2}{r} \cos \gamma \cos \psi \tan \phi \\ & + 2 {}^RV\omega_\oplus (\sin \psi \cos \phi \tan \gamma - \sin \phi) - \frac{r\omega_\oplus^2}{\cos \gamma} \sin \phi \cos \phi \cos \psi \end{aligned} \quad (3.30)$$

Overall, the equations of motion employing the spherical gravity and J_2 -gravity models are given by Eqs. (3.11) – (3.13) and Eqs. (3.28) – (3.30), respectively.

Trajectory Dynamics Model Flow Diagram

The trajectory dynamics model was constructed as a collection of modules comprising the equations of motion, models for the atmosphere, gravity, and TAV, as well as the requisite physical constants from Table 3.3. With this construct, the user is permitted to edit the supporting modules pertaining to the dynamical, environmental, and vehicle models without effecting the operation of the differential equation solver routine encapsulated in the core program. A flow diagram of the trajectory dynamics model with all supporting modules is below.

⁸⁵ Hicks, 413.

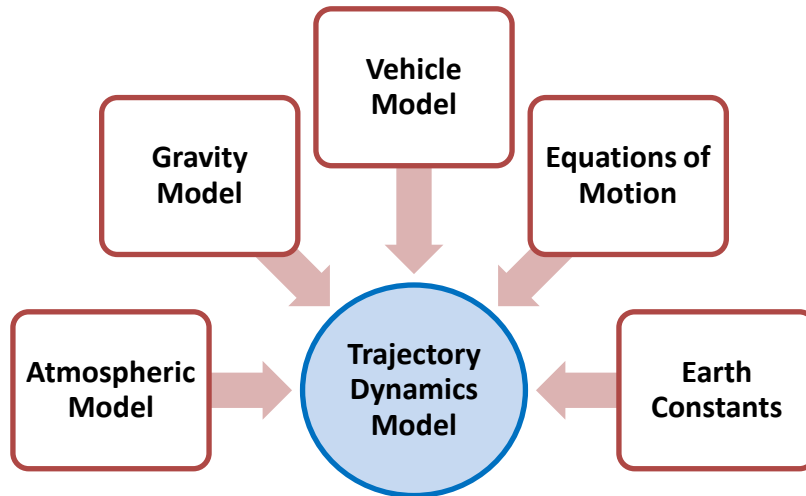


Figure 3.7. Trajectory Dynamics Model Flow Diagram

Model Verification Assumptions

In addition to gravity, the trajectory dynamics model is also reliant on secondary dynamics models related to the planetary atmosphere, planetary angular motion, and TAV aerodynamics. For the purposes of model verification, however, the duplication of the Apollo 10 re-entry profile permits the relaxation of several aforementioned simulation assumptions with the implementation of an exponential density and non-rotating planetary models. In his analysis of the Apollo 10 re-entry in his book *Introduction to Astrodynamic Re-Entry*, Hicks sought to improve his capsule re-entry simulation by replacing the “baseline” exponential density model described by Eq. (3.3) with the 1962 Standard Atmosphere Model. A hypothetical vertical distribution of atmospheric density, pressure, and temperature from sea-level to an altitude of 1000 km, the 1962 Standard Atmosphere Model is an ideal, steady-state representation of the Earth’s atmosphere at a latitude of 45 deg N during “moderate solar activity.”⁸⁶ Compared with the exponential model, the implementation of the 1962 Standard Atmosphere Model revealed

⁸⁶ Vallado, 565.

that the RMS error increased by 0.275%, 8.30%, and 2.34% for the geodetic altitude, inertial velocity, and tangential deceleration solutions, respectively. Although changes to the aerodynamic coefficients could potentially reduce the RMS error associated with the 1962 Standard Atmosphere Model, the implementation of the exponential density model is deemed permissible for purposes of duplicating the Apollo 10 re-entry profile.⁸⁷

Similarly, Hicks also sought to improve his simulation of the Apollo 10 re-entry by including planetary rotation. Since the Apollo 10 initial states are expressed in the inertial frame, a series of coordinate transformations were first completed to convert the states to a frame relative to the rotating Earth. Following the integration of the equations of motion, the Apollo 10 states were then transformed back to the inertial frame.⁸⁸ After simulating the Apollo 10 re-entry with the planetary rotation rate both activated and deactivated, RMS error analysis indicated that the inclusion of planetary rotation created the greatest improvement in accuracy for the inertial velocity solution, while only a “marginal improvement” for geodetic altitude. The specific RMS values for geodetic altitude, inertial velocity, and deceleration are listed in Table 3.4 for the “baseline” case of deactivated planetary rotation as well as for the activated rotation case. When compared with the RMS error for the baseline case, the RMS error for geodetic altitude and velocity improved by 22.4% and 11.6%, respectively, while the RMS error increased by 1.96% for tangential deceleration. With only minor improvements to the trajectory solutions arising from the inclusion of the planetary rotation rate, the assumption of negligible planetary angular motion is also deemed permissible, thus generating the secondary assumption that the initial inertial entry velocity, flight-path angle, and heading angle for Apollo 10 are Earth-relative.⁸⁹

⁸⁷ Hicks, 409, 411.

⁸⁸ Ibid., 393-394.

⁸⁹ Ibid., 383.

Finally, for the capsule aerodynamics model, Hicks chose to represent the drag and lift coefficients for Apollo 10 as constants derived by averaging the preflight aerodynamic coefficient estimates for the Apollo 11 Command Module capsule at Mach 10 and Mach 29.5.⁹⁰ Alternatively, Hicks indicates that the aerodynamic coefficients can also be obtained by first calculating the Mach number as a function of altitude and speed during the integration of the equations of motion, and then continuously adjusting the coefficients by interpolating with the Apollo 11 preflight estimates.⁹¹ After simulating both methods, RMS error analysis revealed that the “baseline” case with constant aerodynamic coefficients produced less error than those derived from the Mach-dependent functions and associated interpolation scheme. In terms of geodetic altitude, the constant and function-derived aerodynamic coefficients produced a RMS error of 3.63 km and 4.48 km, respectively. For inertial velocity, a greater deviation in RMS error is illustrated, with 241 m/s for the constant values and 814 m/s for the function-derived values.⁹² By producing less error than the function-derived aerodynamic coefficients, the assumption of modeling the drag and lift coefficients as constant values is also deemed permissible for the Apollo 10 capsule.

Table 3.4. RMS Errors for Modifications to Trajectory Dynamics Model⁹³

<i>RMS Error Type</i>	<i>Baseline</i>	<i>Modification to Dynamics</i>			
		<i>Gravity (J_2)</i>	<i>Planetary Rotation</i>	<i>Atmosphere</i>	<i>Aerodynamic Coefficients</i>
$RMS_{h_{gd}}$	3.63 km	3.57 km	3.21 km	3.64 km	4.48 km
RMS_{v}	241 m/s	253 m/s	187 m/s	261 m/s	814 m/s
RMS_{decel}	4.60 m/s ²	4.69 m/s ²	5.40 m/s ²	4.71 m/s ²	8.43 m/s ²

⁹⁰ Ibid., 379, 384.

⁹¹ Ibid., 384.

⁹² Ibid., 404.

⁹³ Ibid., 415.

Verification of Trajectory Dynamics Model

Due to the availability of data for both the re-entry initial conditions and trajectory, the Apollo 10 re-entry was chosen as a preliminary means of verifying the trajectory dynamics model described earlier in this chapter. As a method of integrating the equations of motion, a fourth-order Runge-Kutta numerical integrator was employed with the Apollo 10 bank angle history given in Fig. 3.8 as a control input, and the gravitational acceleration described by the J_2 -gravity model.

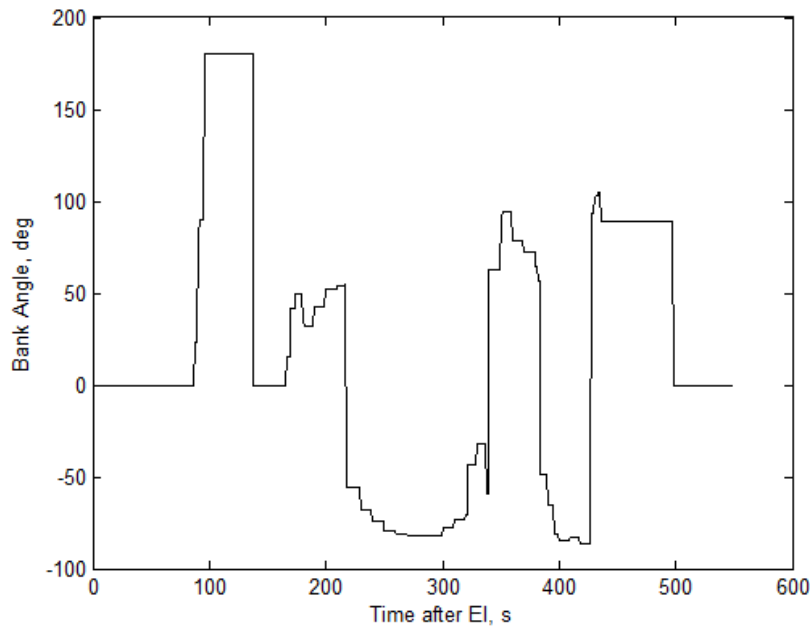


Figure 3.8. Bank Angle History for Apollo 10 Command Module Capsule⁹⁴

Based on the Apollo 10 re-entry solutions obtained from Hicks' text, Fig. 3.9 illustrates that despite initial alignment, the simulated geocentric and geodetic latitude solutions diverge from the Apollo 10 trajectory at approximately 150 s after entry interface (EI), or passage through the upper limit of the sensible atmosphere. Beyond a visual assessment, the divergence exhibited by the simulated latitude solution from the Apollo 10 trajectory can be quantified in

⁹⁴ Ibid., 378.

terms of distance. Due to the ellipticity of the Earth, however, the distance between lines of latitude increases towards the poles and, as a result, cannot be assumed constant. As a result, a sample method for calculating the distance of 1 deg of latitude at specified geocentric latitudes (in units of degrees) is given by the following trigonometric expression:⁹⁵

$$d_{\phi=1 \text{ deg}} = 111.13295 - 0.55982 \cos(2\phi) + 0.00117 \cos(4\phi) \quad (3.31)$$

Employing Eq. (3.31), the approximate distance between the simulated terminal geocentric latitude of 17.1 deg S and the actual value of 15.06 deg S is 221 km. Aside from latitude, an examination of Figs. 3.10 and 3.11 indicate that while the geodetic altitude solution tracks closer to the Apollo 10 trajectory, the inertial velocity solution diverges at 150 s after EI – the same time as indicated by the latitude plot.

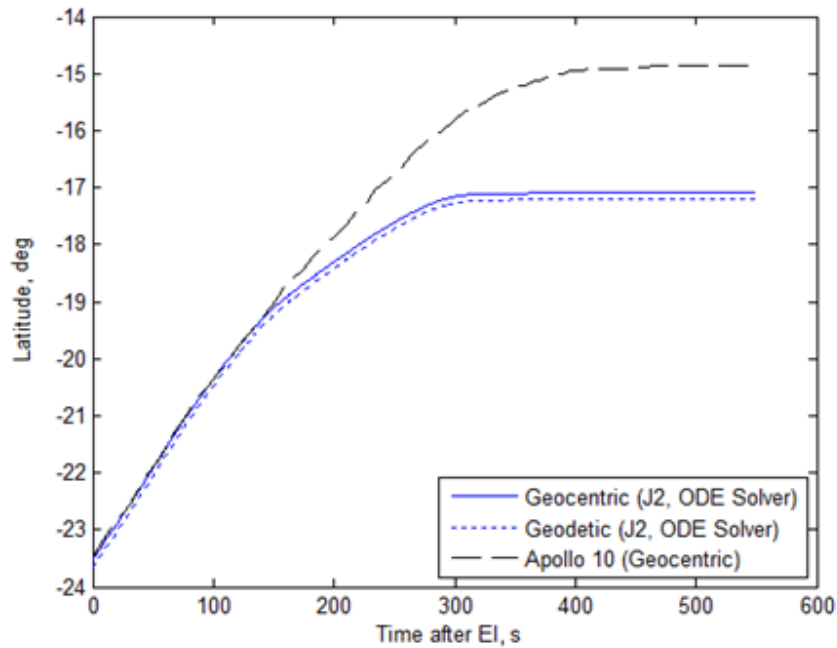


Figure 3.9. Comparison of Geocentric/Geodetic Latitude for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)

⁹⁵ Larry McNish, "Latitude and Longitude," RASC Calgary Centre, The Royal Astronomical Society of Canada, last modified 11 November 2011, accessed 17 August 2012, <http://calgary.rasc.ca/latlong.htm>.

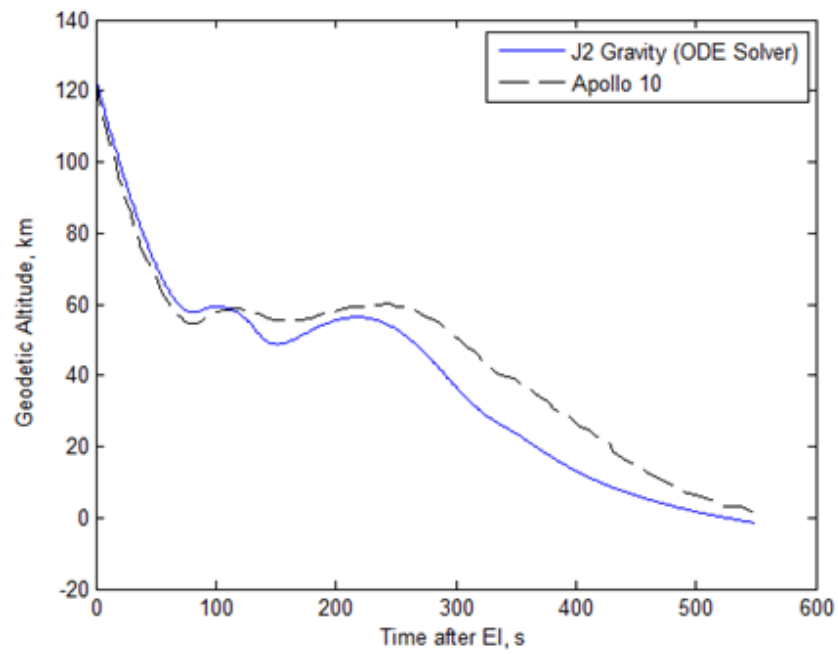


Figure 3.10. Comparison of Geodetic Altitude for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)

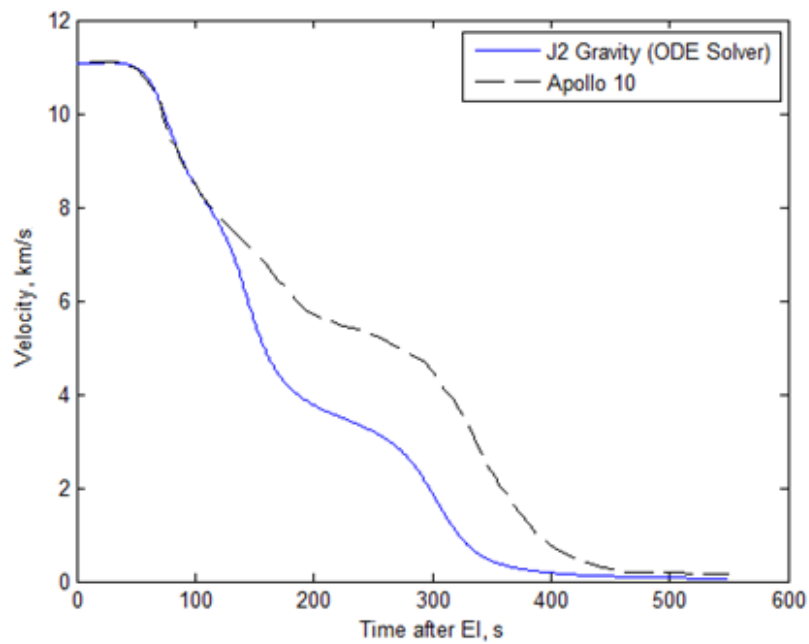


Figure 3.11. Comparison of Inertial Velocity for Apollo 10 (J_2 -Gravity Model, Fourth-Order Runge-Kutta Solver)

For the preceding analysis, the fourth-order Runge-Kutta solver was run with a relative error tolerance of $E_{rel} = 1.0 \times 10^{-8}$ and a default maximum step size of 55 s based on the formula $N_{max} = (0.1 \cdot |t_0 - t_f|)$, where $t_0 = 0$ s and $t_f = 550$ s. After a limited sensitivity analysis run to identify the impact of modifying these parameters on the trajectory solutions, updated values for relative error tolerance and maximum step size were selected to be 1.0×10^{-10} and 0.1 s, respectively. Illustrated in Figs. 3.12–3.14, the modified parameters improved the performance of the fourth-order Runge-Kutta solver for not only the latitude, but also the geodetic altitude and inertial velocity solutions. Quantitatively, the improved solver performance is expressed by RMS error and outlined in Table 3.5. Compared with the initial simulation run, the reduction of both the relative error tolerance and maximum step size produced a respective 69.11%, 70.80%, and 67.61% decrease in the RMS error for the latitude, geodetic altitude, and inertial velocity solutions.

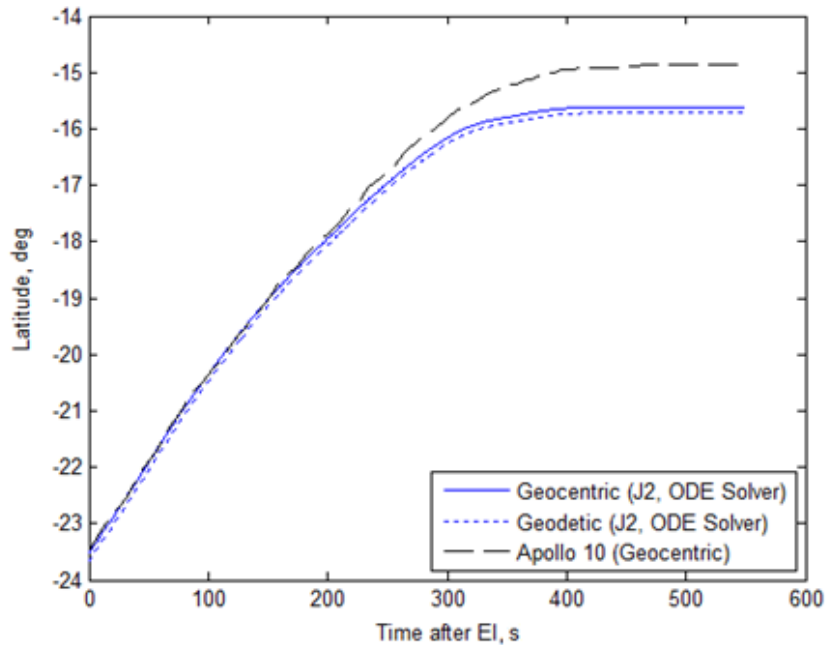


Figure 3.12. Comparison of Geocentric/Geodetic Latitude for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters)

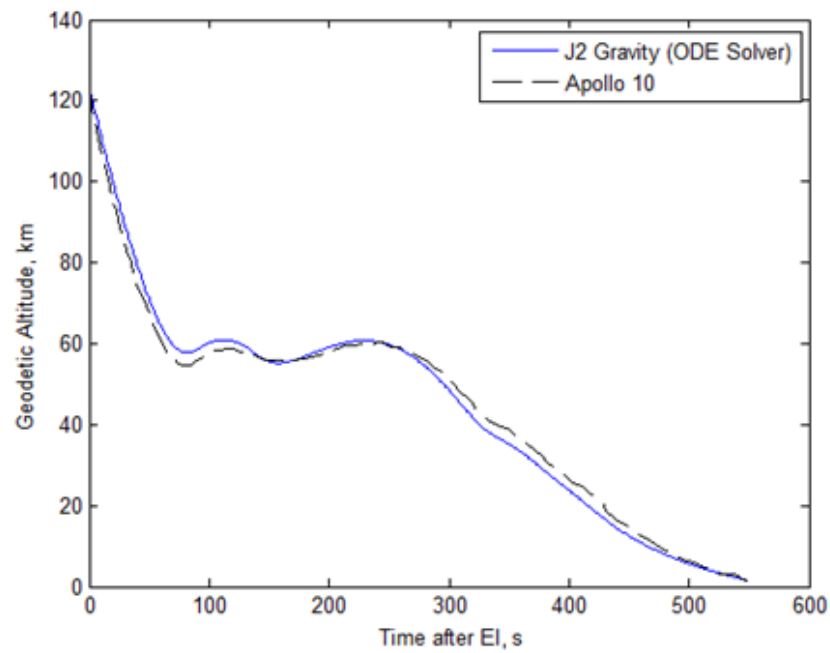


Figure 3.13. Comparison of Geodetic Altitude for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters)

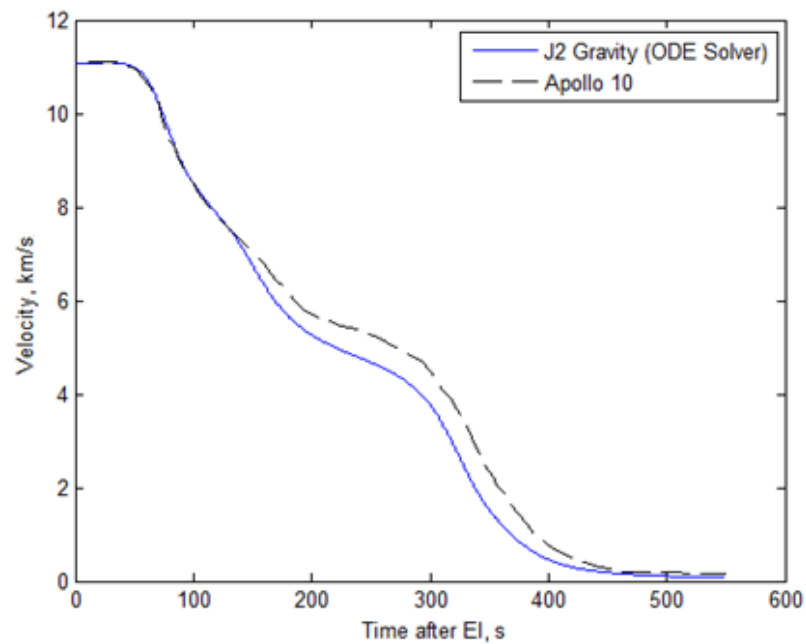


Figure 3.14. Comparison of Inertial Velocity for Apollo 10 (J_2 -Gravity Model, Modified Solver Parameters)

Table 3.5. RMS Error for Trajectory Dynamics Model Verification

	<i>Geocentric Latitude, ϕ</i>	<i>Geodetic Altitude, h_{gd}</i>	<i>Velocity, IV</i>
Total Points, n	56	107	73
Initial Simulation	1.35 deg	9.59 km	1420 m/s
Modified Parameters	0.417 deg	2.80 km	460 m/s

Despite the improvement in RMS error, deviation between the Apollo 10 trajectory and the solutions produced by the trajectory dynamics model persisted in subsequent simulations. With the remaining error resulting from neither incorrect unit conversions nor the erroneous transcription of the equations of motion into the computational software, the Apollo 10 capsule aerodynamic coefficients were next examined and a solution sensitivity study performed. Outlined in Table 3.6, the drag and lift coefficients for the Apollo 10 capsule were modified from their original values of $C_D = 1.2569$ and $C_L = 0.40815$ and simulated with various combinations of relative error tolerance and maximum step size. With the original aerodynamic coefficients obtained by averaging the preflight estimates for the Apollo 11 capsule at Mach 10 and Mach 29.5, the modified values were selected from the aerodynamic coefficients corresponding to the same Mach number range given by:⁹⁶

$$1.2246 \leq C_D \leq 1.2891$$

$$0.38773 \leq C_L \leq 0.42856$$

Due to the complexities of increasing and/or decreasing the aerodynamics coefficients while endeavoring to simultaneously minimize the RMS error for geocentric latitude, geodetic altitude, and inertial velocity, the aerodynamic coefficients listed in Table 3.6 represent optimal estimates. From the various cases analyzed, the alternate aerodynamic coefficients which yielded the lowest RMS error for geocentric latitude, geodetic altitude, and inertial velocity are $C_D = 1.255$ and

⁹⁶ Hicks, 379, 384.

$C_L = 0.4225$ for a relative error tolerance and maximum step size of 1.0×10^{-10} and 0.1 s, respectively. Trajectory solutions corresponding to these aerodynamic coefficients are illustrated in Figs. 3.15-3.17.

Table 3.6. RMS Error for Alternate Aerodynamic Coefficients

<i>Rel. Error Tol., E_{rel}</i>	<i>Max. Step Size, N_{max}</i>	C_L	C_D	<i>Geocentric Latitude, ϕ</i>	<i>Geodetic Altitude, h_{gd}</i>	<i>Velocity, 1V</i>
1.0×10^{-10}	0.1 s	0.40815	1.2569	0.417 deg	2.80 km	460 m/s
1.0×10^{-10}	0.1 s	0.4225	1.255	0.0338 deg	1.11 km	105 m/s
1.0×10^{-10}	0.5 s	0.4240	1.245	0.0406 deg	1.21 km	125 m/s
1.0×10^{-10}	1.0 s	0.4260	1.251	0.0412 deg	1.23 km	123 m/s
1.0×10^{-8}	0.1 s	0.4234	1.257	0.0381 deg	1.18 km	114 m/s
1.0×10^{-8}	0.5 s	0.4240	1.258	0.0442 deg	1.27 km	127 m/s
1.0×10^{-8}	1.0 s	0.4265	1.235	0.0405 deg	1.21 km	128 m/s

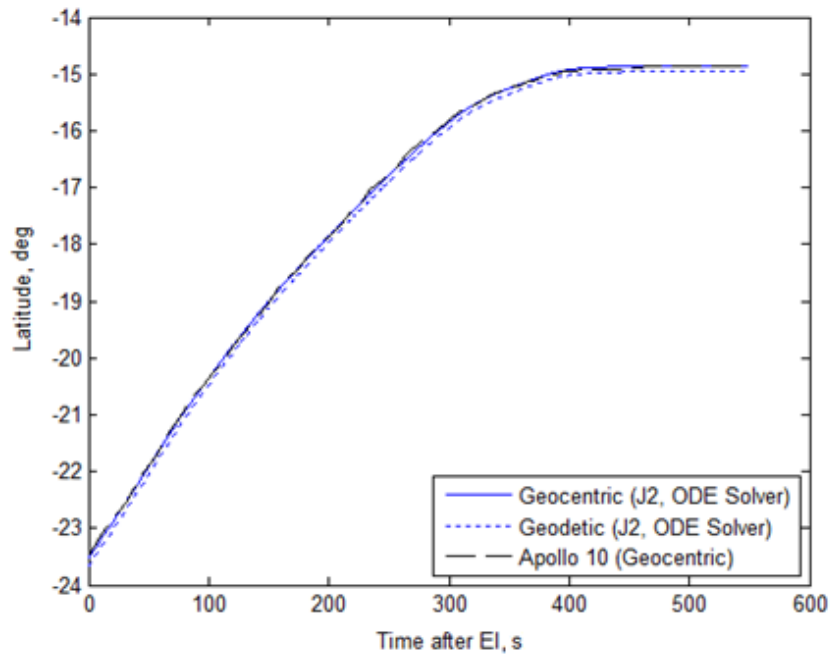


Figure 3.15. Comparison of Geocentric/Geodetic Latitude for Apollo 10
 $(C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1)$

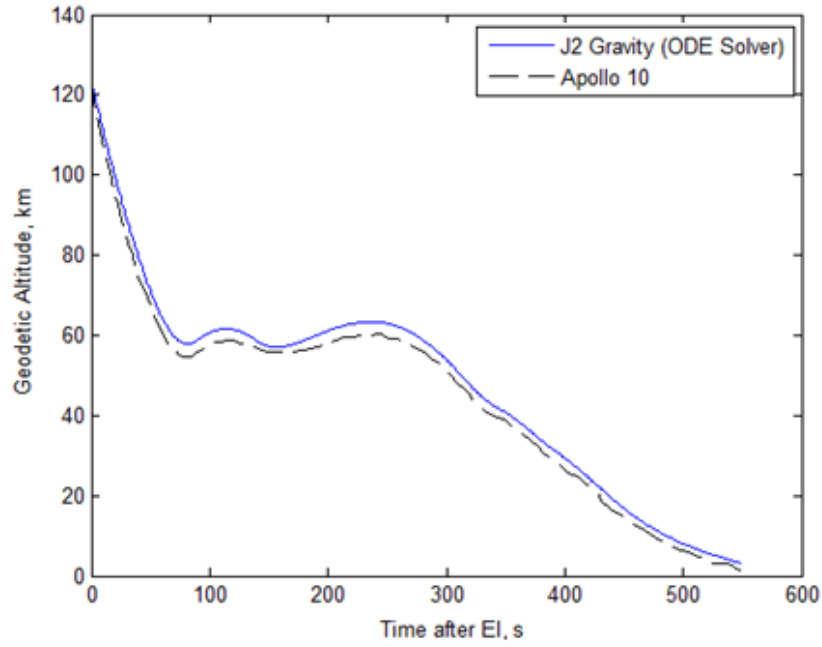


Figure 3.16. Comparison of Geodetic Altitude for Apollo 10 ($C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1$)

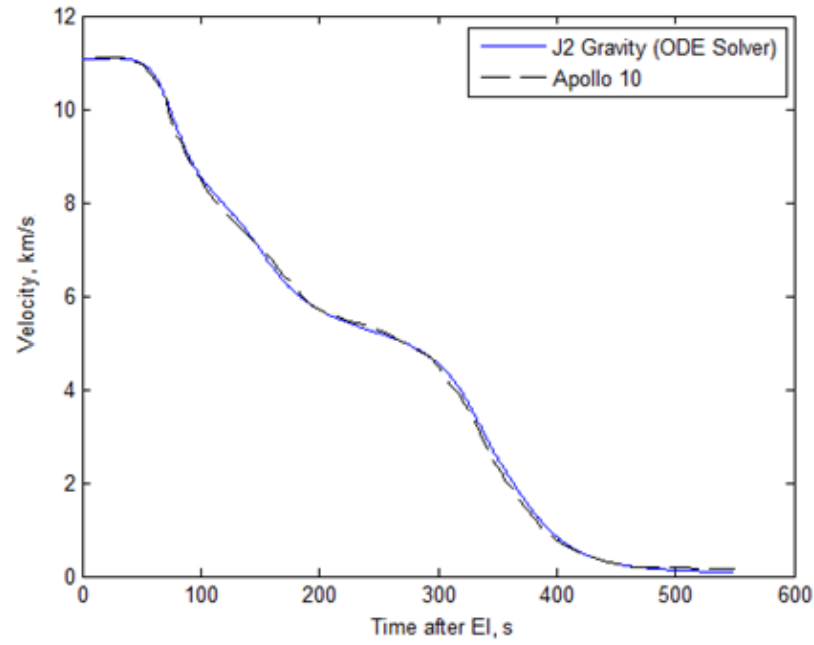


Figure 3.17. Comparison of Inertial Velocity for Apollo 10 ($C_L = 0.4225, C_D = 1.255, E_{rel} = 1 \times 10^{-10}, N_{max} = 0.1$)

Although the modified aerodynamic coefficients yielded trajectory solutions with the hitherto lowest RMS error during the model verification process, such results remain dissonant with Hicks' text since all Apollo 10 re-entry analysis was accurately performed with the original values of $C_D = 1.2569$ and $C_L = 0.40815$. Consequently, the fourth-order Runge-Kutta solver underpinning the trajectory dynamics model was re-examined for sources of possible error beyond the relative error tolerance and maximum step size parameters. Shifting investigative focus towards the solver inputs, it was determined that the solver was interpolating the control input from the one-second incremented bank angle profile given in Fig. 3.8 while simulating the capsule re-entry trajectory with non-integer time steps. Illustrated in Figs. 3.18 and 3.19, the interpolated bank angle profile (shown in red) does not align with the original profile and thus introduces erroneous bank angle values into the simulation.

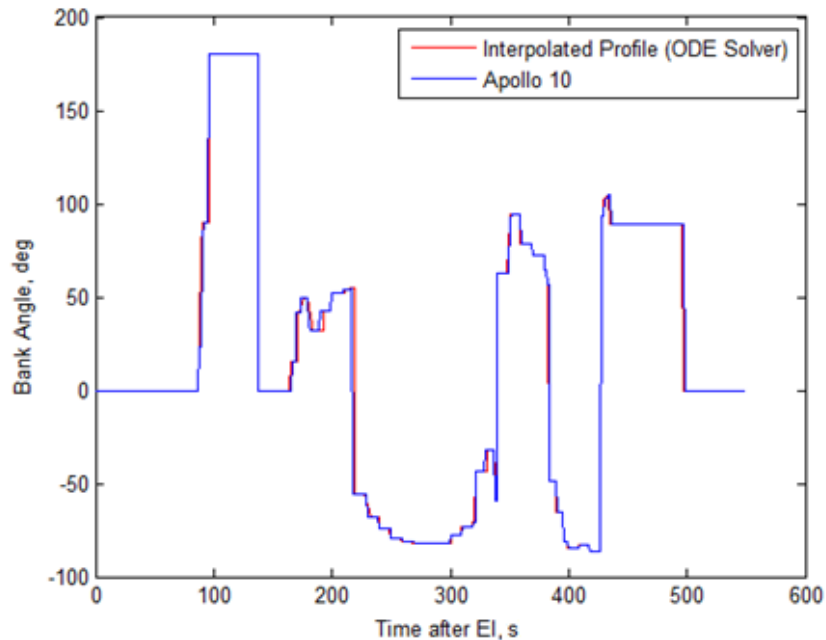


Figure 3.18. Comparison of Bank Angle Profile for Apollo 10 ($C_L = 0.40815$, $C_D = 1.2569$, $E_{rel} = 1 \times 10^{-8}$, $N_{max} = \text{Default}$)

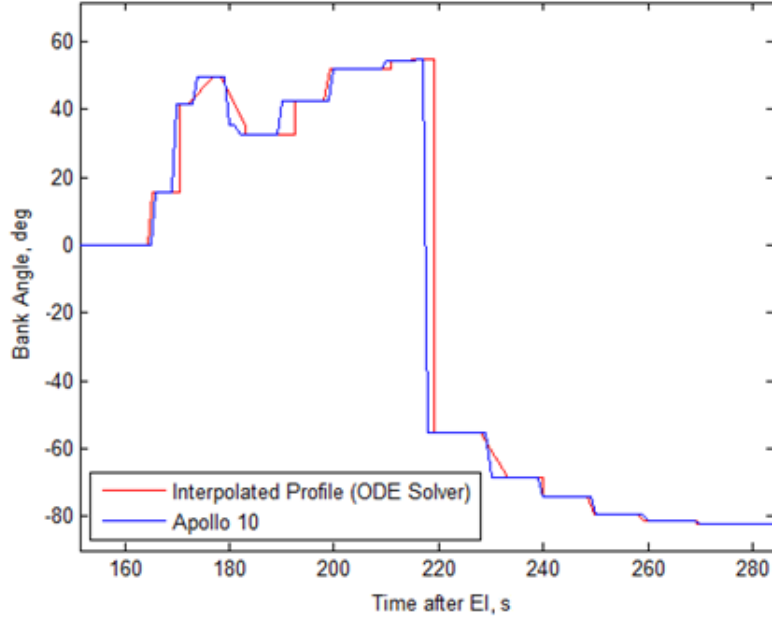


Figure 3.19. Comparison of Bank Angle Profile for $t = [160, 280]$ s
 $(C_L = 0.40815, C_D = 1.2569, E_{rel} = 1 \times 10^{-8}, N_{max} = \text{Default})$

In order to prevent this interpolation, integer rounding code was introduced which forces the time steps to align with the bank angle profile time history, thereby producing the correct control input. When run with the original aerodynamic coefficients of $C_D = 1.2569$ and $C_L = 0.40815$, the trajectory dynamics model produced trajectory solutions with RMS errors of 0.0447 deg, 0.8047 km, and 61.0 m/s for geocentric latitude, geodetic altitude, and inertial velocity, respectively. Plots for trajectory solutions corresponding to the preceding RMS error values are shown by Figs. 3.20 – 3.22.

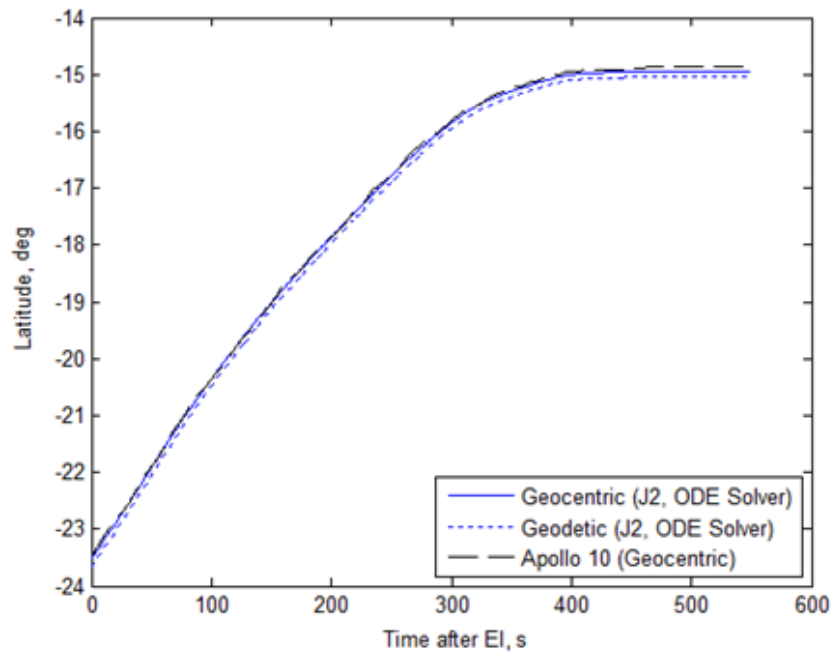


Figure 3.20. Comparison of Geocentric/Geodetic Latitude for Apollo 10 with Non-Interpolation of Bank Angle Profile

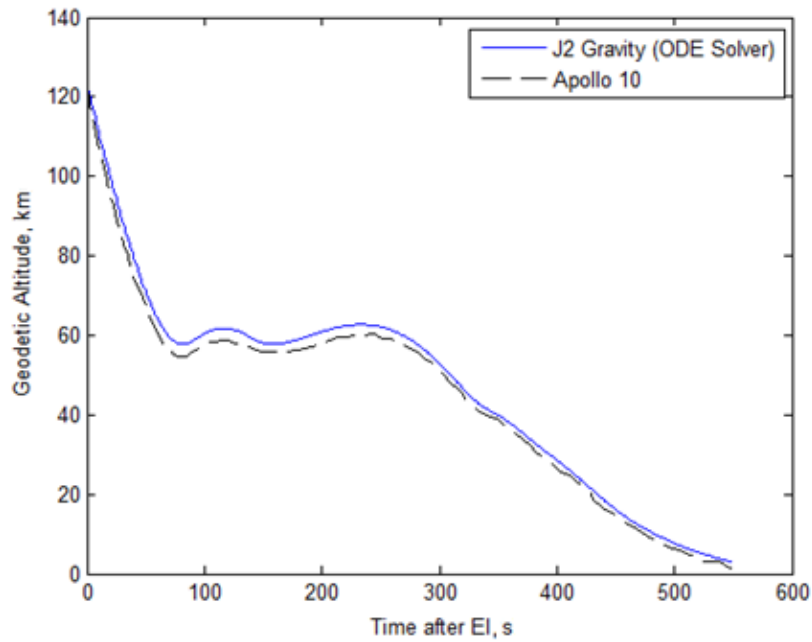


Figure 3.21. Comparison of Geodetic Altitude for Apollo 10 with Non-Interpolation of Bank Angle Profile

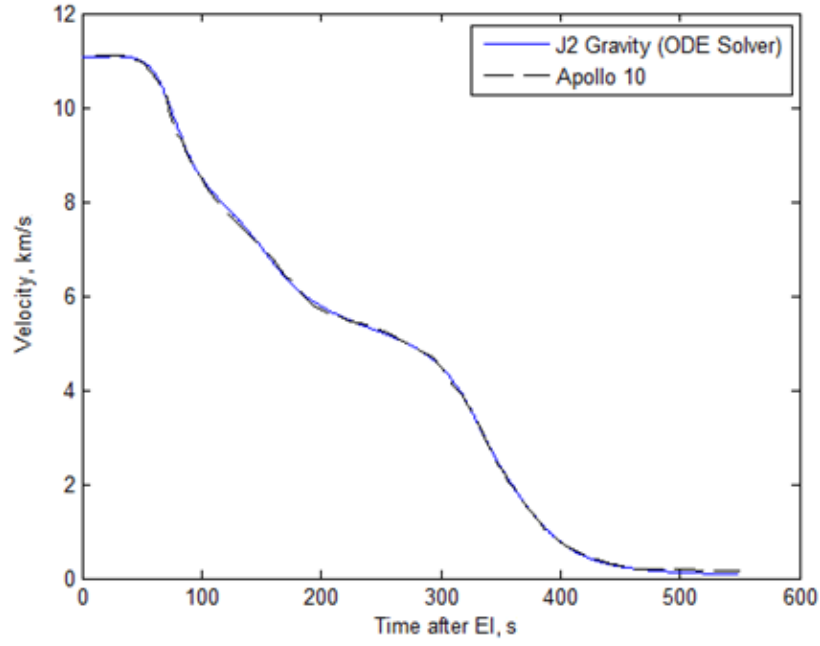


Figure 3.22. Comparison of Inertial Velocity for Apollo 10 with Non-Interpolation of Bank Angle Profile

Verification of Deceleration Model

A model for re-entry deceleration is given by Hicks and provides dimensional values according to:⁹⁷

$$(a_{decel})_V = -{}^R\dot{V} = \frac{D}{m} + g(r) \sin \gamma \quad (3.32)$$

$$(a_{decel})_L = -\dot{\gamma} {}^RV = -\frac{L}{m} - \left[\frac{{}^RV^2}{r} - g(r) \right] \cos \gamma \quad (3.33)$$

$$a_{decel} = \|\bar{a}_{decel}\| = \sqrt{(a_{decel})_V^2 + (a_{decel})_L^2} \quad (3.34)$$

where $(a_{decel})_V$ and $(a_{decel})_L$ are the tangential (along the velocity vector) and normal (along the lift vector) components of the deceleration vector, respectively. When divided by the

⁹⁷ Hicks, 66.

acceleration due to gravity at a specified reference altitude, g_0 , then the deceleration components and overall magnitude calculated in Eqs. (3.32) – (3.34) become non-dimensional quantities.

When simulating the Apollo 10 re-entry deceleration profile for a spherical gravity and rotating planetary model, the preceding equations yield the non-dimensional solutions illustrated in Fig. 3.23. Although over-estimating the local maxima of the Apollo 10 profile, the deceleration model produces solutions which coincide with the general locations for the profile maxima and minima over the specified time-of-flight. In terms of RMS error, the visual assessment of the model's graphical behavior translates into a 0.422 g-deviation of the tangential component from the Apollo 10 profile, and 0.578 g for the deceleration magnitude.

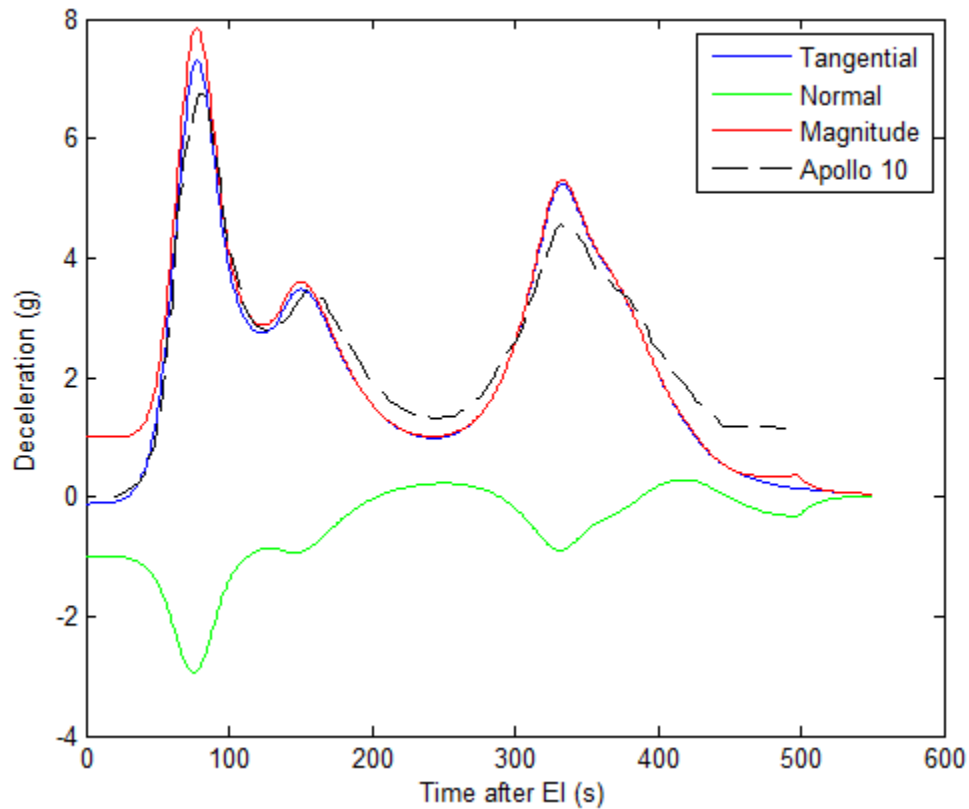


Figure 3.23. Comparison of Deceleration for Apollo 10 with Spherical Gravity and Rotating Planetary Models

Verification and Selection of Heat Flux Model

As with the verification of the trajectory dynamics model, measurements and simulation solutions available from various NASA missions were utilized to verify the efficacy of the stagnation heat flux models presented in Chapter II. Once verified, an appropriate model was selected and applied to all aeroassisted maneuver simulations. Although unavailable for the Apollo 10 capsule, re-entry heat flux data was obtained for two sub-orbital Apollo command module flights performed in February and August 1966.⁹⁸ The vehicles employed for the tests, identified as Apollo Spacecraft 009 and 011, were fitted with pressure transducers and surface-mounted calorimeters according to Fig. 3.24.

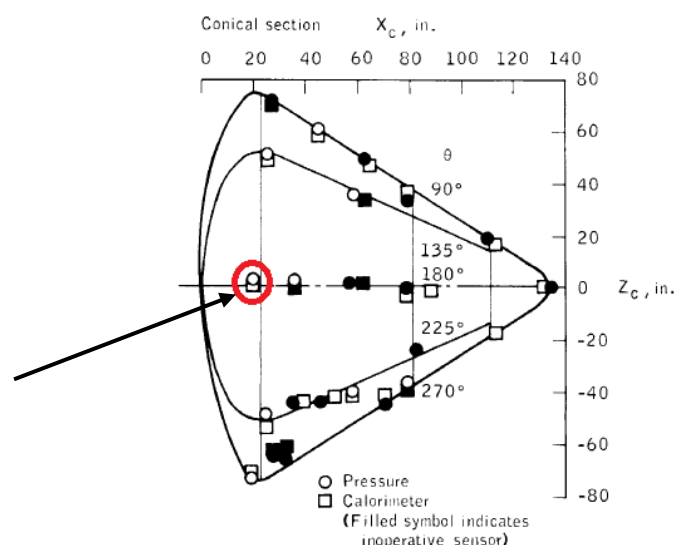


Figure 3.24. Pressure Transducer and Calorimeter Locations on the Conical Section of Apollo Spacecraft 009⁹⁹

⁹⁸ Dorothy B. Lee, John J. Bertin, and Winston D. Goodrich, "Heat-Transfer Rate and Pressure Measurements Obtained during Apollo Orbital Entries," *NASA TN D-6028* (Washington, D.C.: National Aeronautics and Space Administration, 1970), 1.

⁹⁹ *Ibid.*, 17.

For Spacecraft 009, local pressure and heat flux measurements were obtained at free-stream relative velocities between $12,000 \leq V \leq 25,300$ ft/s ($3.7 \leq V \leq 7.7$ km/s); for Spacecraft 011, the velocity range was greater at $2,080 \leq V \leq 27,300$ ft/s ($0.63 \leq V \leq 8.3$ km/s).¹⁰⁰ At an altitude and velocity of 45.7 km and 6.86 km/s at 64 s after EI, Spacecraft 009 achieved a maximum measured heat flux of 210 kW/m² at the calorimeter location identified by the circle and arrow in Fig. 3.24. For the same calorimeter location, Spacecraft 011 achieved a maximum heat flux of 94 kW/m² at an altitude and velocity of approximately 64.0 km and 7.62 km/s at 170 s after EI.¹⁰¹ Since “no valid heat transfer data” was obtained on the blunt entry face of either command module, no real depiction of the heat flux immediately behind the bow shock is available.¹⁰² Consequently, the aforementioned heat flux measurements constitute the only maximum values available for Spacecraft 009 and 011 in subsequent comparative analysis.

In addition to the Apollo sub-orbital flights, heat flux data also exists for the Space Shuttle, specifically the STS-5 (Space Transport System) mission of 1982. Shown in Fig. 3.25, thermocouples were affixed within sections of the fuselage and wings so as to enable the measuring of total heat flux at varying locations relative to both the vehicle centerline and hypersonic flow. Overall, the fuselage sidewalls, cargo bay doors, and the upper wing surfaces are subject to lower heating rates, while the fuselage and wing lower surfaces are conversely subject to higher heating rates.¹⁰³

¹⁰⁰ Ibid., 1.

¹⁰¹ Ibid., 21, 24, 52, 69.

¹⁰² Ibid., 1.

¹⁰³ William L. Ko, Robert D. Quinn, and Leslie Gong, “Finite Element Re-Entry Heat Transfer Analysis of Space Shuttle Orbiter,” *NASA TP 2657* (Edwards, CA: NASA Dryden Flight Research Facility, 1986), 1.

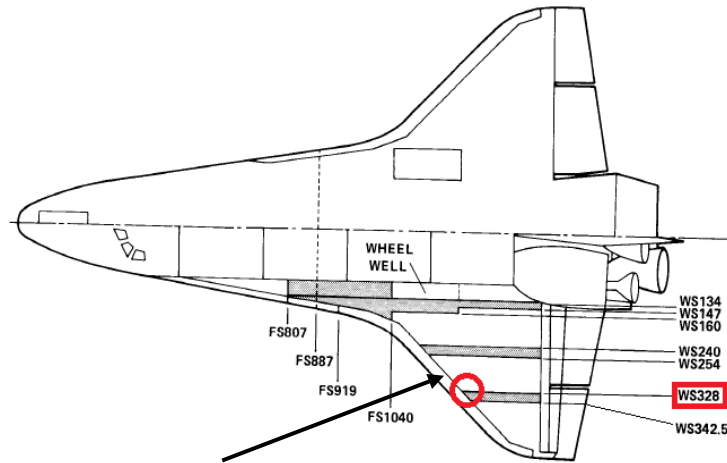


Figure 3.25. Wing Segment (WS) and Fuselage Section (FS) Locations used for STS-5 Heat Flux Analysis¹⁰⁴

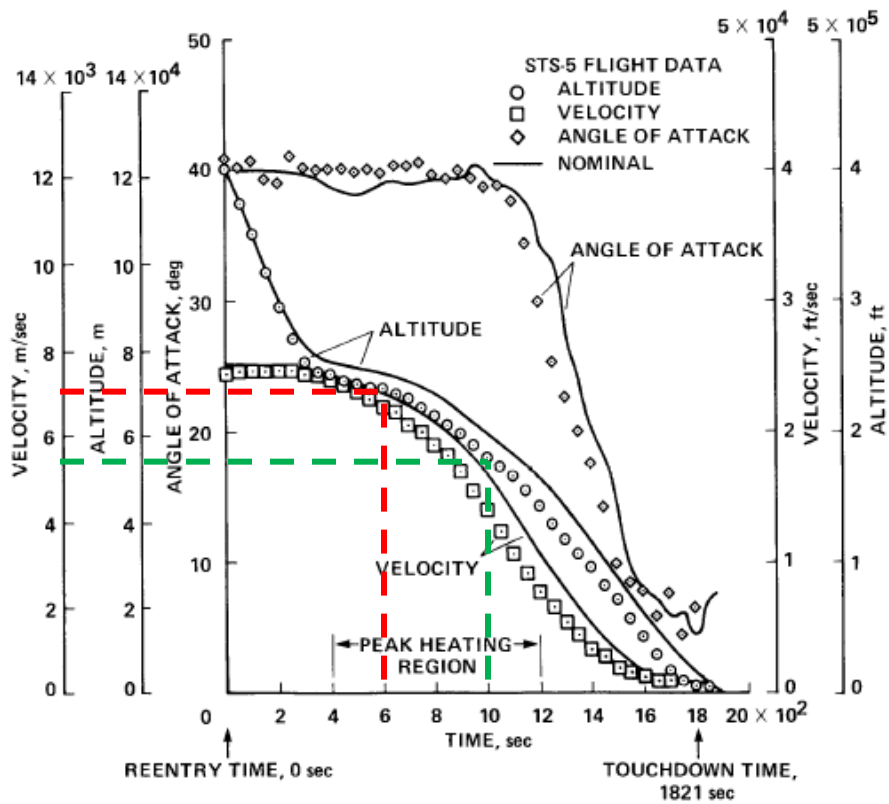


Figure 3.26. Re-Entry Trajectory for STS-5¹⁰⁵

¹⁰⁴ Ibid., 16, 18.

¹⁰⁵ Ibid., 16.

As presented in their 1986 paper “Finite Element Re-Entry Heat Transfer Analysis of Space Shuttle Orbiter,” Ko, Quinn, and Gong indicate that the maximum heat flux was measured to be approximately 1400 kW/m^2 and occurred on the lower surface of Bay 1 in WS328 (circle and arrow in Fig. 3.25) at 600 s after EI.¹⁰⁶ Based on the trajectory profile for STS-5 given in Fig. 3.26, this time corresponds to an approximate altitude and velocity of 70 km and 7.0 km/s, respectively. While also occurring on the lower surface of Bay 1 in WS328, the peak temperature of 1910°F was measured at a later elapsed time of 1000 s at a lower approximate altitude 55 km.¹⁰⁷ The maximum heat flux and peak temperature are respectively denoted by the red and green dashed lines in Fig. 3.26.

A final source of heat flux data originates from a 2007 technical memorandum entitled “Re-Entry Thermal Analysis of a Generic Crew Exploration Vehicle Structure.” In this paper, Ko, Gong, and Quinn utilized an organic NASA Dryden aerodynamic heating software program to calculate the heat flux encountered by the Crew Exploration Vehicle (CEV) when flying the identical trajectory as Apollo Spacecraft 009 in 1966.¹⁰⁸ Initially, the zero-tilt stagnation heat flux was calculated and featured a maximum value of 818 kW/m^2 at an altitude of 45.7 km and an elapsed time of 1630 s. Utilizing the zero-tilt data, an amplification factor of 1.4 was applied to simulate the migration of the stagnation point “toward the upper torodial shoulder” when the CEV is at an 18 deg angle of tilt. Based on this modification, the maximum stagnation heat flux increased to 1128 kW/m^2 .¹⁰⁹

¹⁰⁶ Ibid., 32.

¹⁰⁷ Leslie Gong, William L. Ko, Robert D. Quinn, and W. Lance Richards, “Comparison of Flight-Measured and Calculated Temperatures on the Space Shuttle Orbiter,” *NASA TM 88278* (Edwards, CA: NASA Dryden Flight Research Facility, 1987), 36.

¹⁰⁸ William L. Ko, Leslie Gong, and Robert D. Quinn, “Re-Entry Thermal Analysis of a Generic Crew Exploration Vehicle Structure,” *NASA TM 2007-214607* (Edwards, CA: NASA Dryden Flight Research Facility, 2007), 9.

¹⁰⁹ Ibid., 9, 41-42.

Of the stagnation heat flux models presented in Chapter II, only four were selected for comparison with the preceding flight data for Apollo, the Space Shuttle, and CEV: (1) Eq. (2.4) from Darby and Rao (2010); (2) Eq. (2.4) from Rao *et al.* (2002); (3) Eq. (2.2) from Havey; and (4) Eq. (2.6) from Galman. Excluded from consideration, the Detra *et al.* model given in Eq. (2.1) requires presently unknown quantities for stagnation and wall enthalpies, while the form of Eq. (2.4) presented in the Rao *et al.* paper “A Concept for Operationally Responsive Space Mission Planning Using Aeroassisted Orbital Transfer” maintains a coefficient that is four orders of magnitude smaller than both the 2002 and 2010 alternatives. When simulated by the preceding models, the sample NASA vehicle trajectory states corresponding to maximum heat flux produce the results illustrated in Fig. 3.27.

Depicted as a series of colored bars, the models show perceivable variation with the flight data, presented by the cross-hatched bars. For the Apollo spacecraft 009, the models over-estimate the heat flux by one order of magnitude, with the Darby and Rao, and Rao *et al.* variants yield approximately 5200 kW/m², compared with that of 210 kW/m² from the flight data. Similarly, the models over-estimate the heat flux for the Apollo spacecraft 011, but by nearly two orders of magnitude. While the variation with the CEV data is less than that of the Apollo spacecraft, the models still over-estimate the heat flux by 536% compared with the zero-tilt CEV and an associated data amplification factor of 1.0. Conversely, the models under-estimate the heat flux for STS-5 with the Darby and Rao, and Rao *et al.* variants producing a value of approximately 715 kW/m², 48.9% less than the measured 1400 kW/m².

Although patently inaccurate in their estimation the flight data maximum heat flux, several factors must be considered when verifying the efficacy of the respective models. First, the greatest variation between the model and flight data heat flux occurs when the latter

corresponds to a blunt-body spacecraft like that of the capsule design for both the Apollo command module and CEV. Second, the models only estimate stagnation heat flux and do not account for radiative heat flux contributions to the total heat load. While the addition of a radiative heat flux estimate would further increase the variation between the models and Apollo/CEV flight data, it would decrease the variation with the STS-5 flight data and produce an improved approximation of maximum heat flux. Third, the models were empirically formulated primarily with heat flux measurements from experimental devices such as shock tubes located at sea-level. Even though an expedient substitute for flight data, shock tubes and similar devices fail to accurately simulate hypersonic flow effects stemming from not only altitude and varying atmospheric density due to local solar conditions, but also the intermolecular reaction and energy transfer properties of atmospheric atomic and molecular species local to the spacecraft.

Aside from their inherent inaccuracies, the models still provide a coarse approximation of heat flux, with the least variation illustrated with flight data from STS-5, a winged-entry vehicle similar to the example TAV utilized for this research. Overall, the Rao *et al.* 2002 model (referred hereafter as *Rao, 2002*) will be implemented henceforth since it maintains a comparatively small variation with the STS-5 example, as well as a traceable formulation lineage to the experimental work of Detra *et al.*

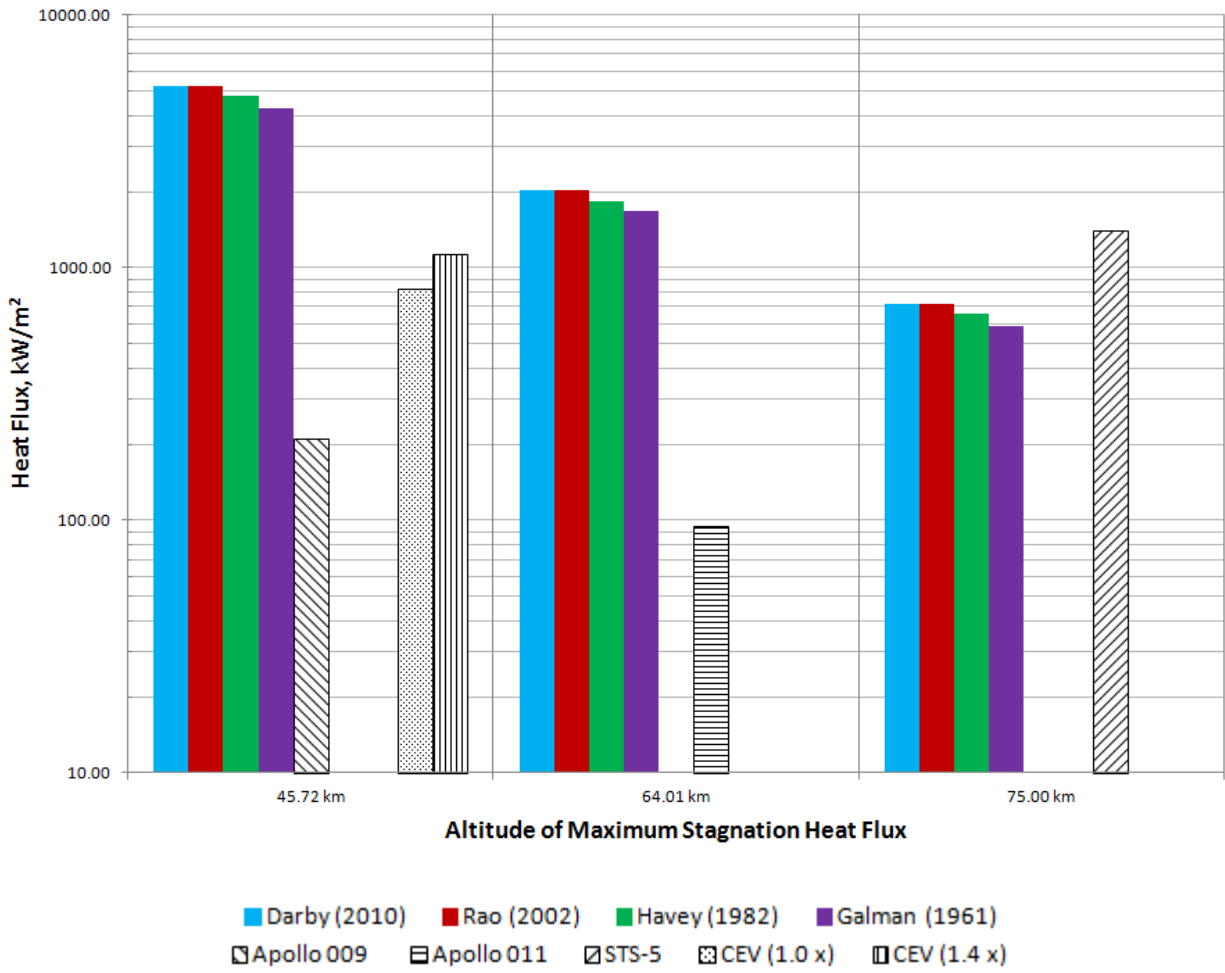


Figure 3.27. Comparison of Stagnation Heat Flux Models with Flight Data from Sample NASA Vehicles

Summary and Conclusion

Selected as a known example of atmospheric re-entry, the Apollo 10 trajectory was duplicated by an independently-developed trajectory dynamics model, thereby verifying the accuracy and efficacy of the model for the simulation of exo- and trans-atmospheric maneuvers. Driven by a system of six differential equations of motion, the trajectory dynamics model is comprised of environmental and planetary models for atmospheric density, gravitational potential, and stagnation heat flux. Rather than utilizing a single model for atmospheric density, a piecewise-continuous atmospheric density function has been developed which models the MSIS-E-90 density profile by incorporating three separate altitude-delimited models. Based on the simulation assumptions of spherical planetary geometry and negligible radiative heat flux contributions during re-entry, the remaining components of the trajectory dynamics model are represented by a spherical gravity model and an empirically-derived model for stagnation heat flux.

IV. Comparative Study of Phasing, Skip Entry, and Simple Plane Change Maneuvers

Chapter Overview

A suite of maneuvers comprising planar phasing, out-of-plane skip entry, and simple plane changes are simulated for a notional trans-atmospheric, lifting re-entry vehicle with $L/D = 6$. By comparing the relative performance of each maneuver to overfly a geographically diverse sample ground targets, it is demonstrated that skip entry maneuvers require a total ΔV less than 0.5 km/s. For select targets, simulation results demonstrate a significant savings in ΔV expenditure for skip entry compared with the simple plane change alternative. Overall, the simulated skip entry maneuvers consistently provide responsive mission execution in terms of ground target time-of-arrival, with maximum deceleration and stagnation heat flux less than 1.0 g and 1000 kW/m², respectively.

Introduction

Defined as a special case of lifting entry, a skip entry maneuver is comprised of exo- and trans-atmospheric trajectory segments as described by the example in Fig. 4.1. For the present research, the sequence of maneuver events for skip entry commences with a de-orbit impulse applied by the TAV at an initial circular orbit altitude, h_i (A). By decreasing orbital velocity, the initial circular orbit – or *reference orbit* – is transformed into an elliptical orbit with apogee equal to h_i , and perigee corresponding to the desired depth of atmospheric penetration. Following (A), orbital altitude decreases until perigee transit at (B), which occurs below the upper limit of the sensible atmosphere at an altitude of approximately 120 km. During the trans-atmospheric trajectory segment, the TAV generates and utilizes atmospheric lift to execute an out-of-plane maneuver by banking left or right.

As the altitude of a skip entry trajectory decreases, the TAV encounters increasing atmospheric density and, therefore, greater aerodynamic drag. In the absence of drag, the TAV states at (A) would equal those at the end of the trans-atmospheric trajectory, or skip apogee (C). By converting kinetic energy into heat, aerodynamic drag reduces both the altitude and velocity of the TAV such that: (1) The skip apogee altitude is less than the initial altitude; and (2) the velocity is less than the orbital velocity at skip apogee. Without performing a re-circularization burn at skip apogee to establish a stable circular orbit, the TAV will re-enter the atmosphere and continue on a phugoid trajectory of decreasing energy and altitude until planetary impact. With the completion of the re-circularization impulse at (C), however, the TAV enters a new circular orbit (D) which is then maintained until the next maneuver is performed, whether exo- or trans-atmospheric in design.

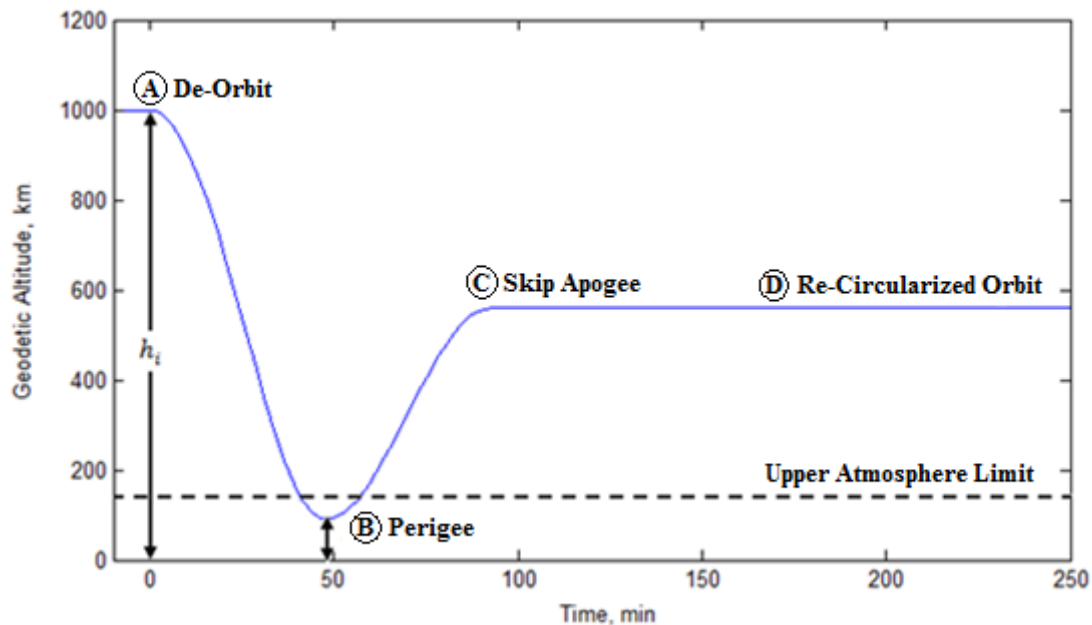


Figure 4.1. Skip Entry Maneuver Diagram

Presented as an alternative to both skip entry and simple plane change maneuvers, planar phasing maneuvers can fulfill a desired mission tasking by either increasing or decreasing the semi-major axis of the reference orbit. With the former case, orbital velocity is increased to create an elliptical orbit with perigee equal to h_i ; for the latter case, orbital velocity is decreased to create an elliptical orbit with apogee equal to h_i and a perigee altitude greater than 120 km, thus precluding any transit through the sensible atmosphere. Perigee placement near the sensible atmosphere limit will, however, yield aerodynamic effects sufficient to degrade the phasing maneuver trajectory if successive perigee transits are executed. Despite such potential effects, the bank angle for all planar phasing maneuvers will remain at $\sigma = 0$ deg.

Methodology

For the planar phasing and out-of-plane skip maneuvers, algorithms are developed to achieve over-flight of a specified ground target by either increasing or decreasing the semi-major axis of an initial reference orbit, or by banking a TAV within the sensible atmosphere to create a plane change. As a means of evaluating the effectiveness of the phasing and skip maneuvers in terms of ground target time-of-arrival and total ΔV , an algorithm is also developed for simple plane change maneuvers conducted in the vacuum environment.

Simulation of Planar Phasing Maneuvers

As an alternative to the exo-atmospheric simple plane change, a TAV can perform either a planar phasing or out-of-plane skip maneuver to fulfill a baseline example of a responsive space mission: Overfly a specified ground target in minimum time. In order to demonstrate the implementation of these maneuver cases, the sample ground target of Moscow was selected due to its mid-latitude location in the Northern Hemisphere. The geographical coordinates for

Moscow along with other sample ground targets utilized for subsequent comparative analyses of maneuver performance are given in Table 4.1. To ensure coverage of all sample ground targets, an initial inclination angle of 70 deg was chosen since it is greater than the latitude of Reykjavik, the northernmost sample location. All remaining reference orbit states are outlined in Table 4.2.

Table 4.1. Geographical Coordinates of Sample Ground Targets of Interest

<i>Ground Target</i>	<i>Longitude</i>	<i>Geodetic Latitude</i>
Reykjavik, Iceland	21.9333 deg E	64.1333 deg N
Moscow, Russia	37.6178 deg E	55.7517 deg N
Tokyo, Japan	139.767 deg E	35.6833 deg N
Gibraltar, United Kingdom	5.3530 deg W	36.1430 deg N
Pontianak, Indonesia	109.333 deg E	0.0000 deg N
Brasilia, Brazil	47.9196 deg W	15.7810 deg S
Buenos Aires, Argentina	58.3817 deg W	34.6036 deg S
Canberra, Australia	149.131 deg E	35.2828 deg S
Cape Town, South Africa	18.4244 deg E	33.9767 deg S

Table 4.2. Reference Orbit Initial States for Over-Flight Analysis

Eccentricity, e	0.0
Altitude, h_i	1000 km
Longitude, θ_i	0 deg
Latitude, ϕ_i	0 deg
Inclination, i_i	70 deg
Flight-Path Angle, γ_i	0 deg
Heading Angle, ψ_i	70 deg
Bank Angle, σ_i	0 deg

As a consequence of simulating trajectories with respect to a rotating planetary reference frame, both the heading angle and orbital velocity need to be recomputed as relative quantities. Defined by the initial reference orbit states, the initial guess for the relative orbital velocity, ${}^R V$, is calculated by utilizing the ${}^R V \dot{\gamma}$ trajectory force equation components as inputs to the quadratic formula when $\dot{\gamma} = 0$:

$${}^R V_n = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (4.1)$$

where the variables a , b , and c are given by:

$$\begin{aligned} a &= \frac{1}{r} \cos \gamma + \left(\frac{\rho C_L S}{2m} \right) \cos \sigma \\ b &= 2\omega_{\oplus} \cos \phi \cos \psi \\ c &= -g(r) \cos \gamma + r\omega_{\oplus}^2 \cos \phi (\cos \phi \cos \gamma - \sin \phi \sin \psi \sin \gamma) \end{aligned}$$

With the computed value for orbital velocity, the initial guess for the heading angle is determined via the *Law of Sines*. When rotation is activated, the heading angle becomes a function of orbital velocity relative to the rotating reference frame, ${}^R V$. Based on the vector geometry of Fig. 4.2, the Law of Sines is employed to produce:

$$\frac{\|{}^R \vec{V}\|}{\sin(\Delta\phi)} = \frac{\|\vec{V}_{\phi_1}\|}{\sin(\psi - \Delta\phi)} = \frac{86400 {}^R V}{\sin(\Delta\phi)} = \frac{2\pi r_i}{\sin(\psi - \Delta\phi)}$$

When algebraically re-arranged, the preceding expression becomes:

$$\psi = \Delta\phi + \sin^{-1} \left(\frac{2\pi r_i \sin(\Delta\phi)}{86400 {}^R V} \right) \quad (4.2)$$

where ${}^R V$ is the orbital velocity calculated from Eq. (4.1). Placed in an iterative loop, Eqs. (4.1) and (4.2) produce relative orbital state solutions when a specified error tolerance is surpassed between the n and $(n - 1)$ steps of the relative heading angle solution algorithm.

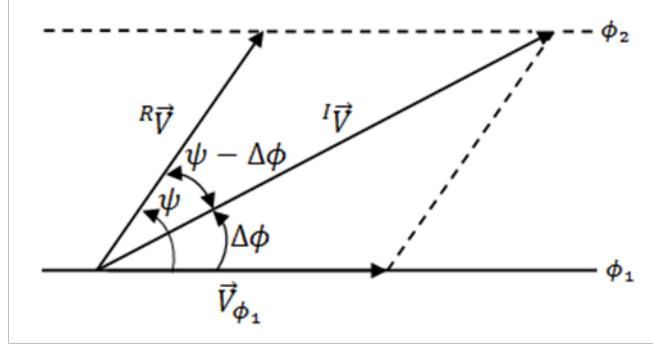


Figure 4.2. Heading Angle, Orbital Velocity with Respect to a Rotating Reference Frame

With the initial values for the relative heading angle and orbital velocity calculated for the TAV, the reference orbit is then propagated for a simulation time of 24 hours. Comparing the final value for semi-major axis with that of the initial state revealed an increase of 0.360836 m and a likewise increase in eccentricity from 0.0 to 2.445×10^{-8} . To ensure that all planar phasing maneuvers commence from a circular reference orbit, the secant iteration method was implemented rather than the traditional Newton-Raphson method. A one-dimension root-finding routine, Newton-Raphson requires the evaluation of both the function $f(x)$ and derivative $f'(x)$ at a point x . Overall, quadratic convergence is achieved by extrapolating the local derivative and geometrically extending a tangent line formed at the current point x_n until it crosses zero, where the next guess x_{n+1} is set equal to the functional value associated with the tangent line zero-crossing, also known as the ordinate.¹¹⁰

For the problem of ensuring that the reference orbit is indeed circular, the heading angle and orbital velocity states must be iteratively calculated so that the difference between the target and post-simulation semi-major axis are within a specified error tolerance. Since functional relationships and their associated derivatives are not readily available for these parameters, the

¹¹⁰ William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge, United Kingdom: Cambridge University Press, 1988), 254-255.

Newton-Raphson method cannot be directly utilized without approximating the derivative $f'(x)$. Alternatively, the derivative requirement can be bypassed by instead using a secant line passing through two points on the curve to iteratively find the root. In terms of orbital velocity, V , and semi-major axis, a , the secant method can be represented as:¹¹¹

$$V_{n+1} = V_n + \frac{(V_n - V_{n-1})(a_0 - a_n)}{a_n - a_{n-1}} \quad (4.3)$$

where the index 0 represents the target condition for semi-major axis.

Following two iterations, the relative values for heading angle and orbital velocity calculated were then used to re-propagate the reference orbit to create a trajectory with a semi-major axis deviation of 0.1804 m – the result of accumulated numerical errors in the differential equation solver. From the ground track trajectory produced by the propagated reference orbit, the approximate locations where the trajectory crossed the line of latitude for the ground target were identified and catalogued. Since the solver produces discrete solutions, the exact longitude corresponding with each latitude crossing cannot be directly determined from the trajectory and, therefore, must be interpolated. Selecting cubic spline rather than a linear interpolation scheme due to the nonlinearities of the trajectory, the longitude of each crossing was calculated and then differenced with the target longitude to produce a “delta”-longitude, or $\Delta\theta$. Figure 4.3 illustrates the ground track trajectory of the propagated reference orbit with respect to the example target of Moscow, while Fig. 4.4 depicts the latitude crossings, interpolation points, and resulting longitude interpolation solutions.

¹¹¹ James F. Epperson, *An Introduction to Numerical Methods and Analysis* (Hoboken, NJ: John Wiley & Sons, Inc., 2007), 120-121.

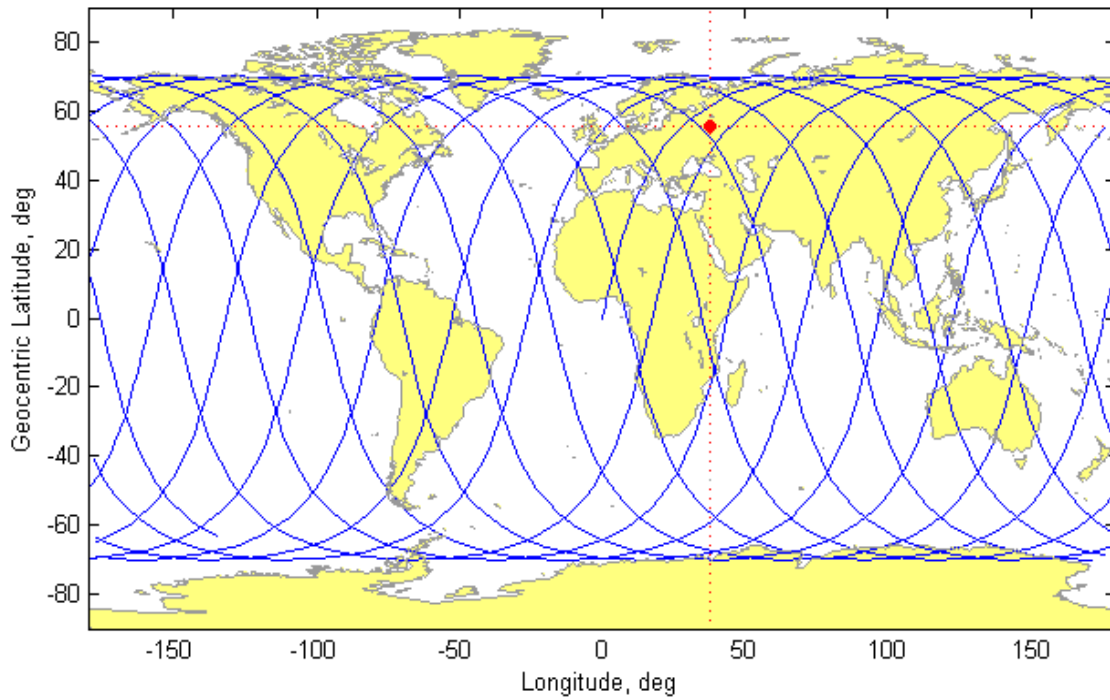


Figure 4.3. Ground Track Trajectory of Reference Orbit
($h_i = 1000$ km, $i_i = 70$ deg)

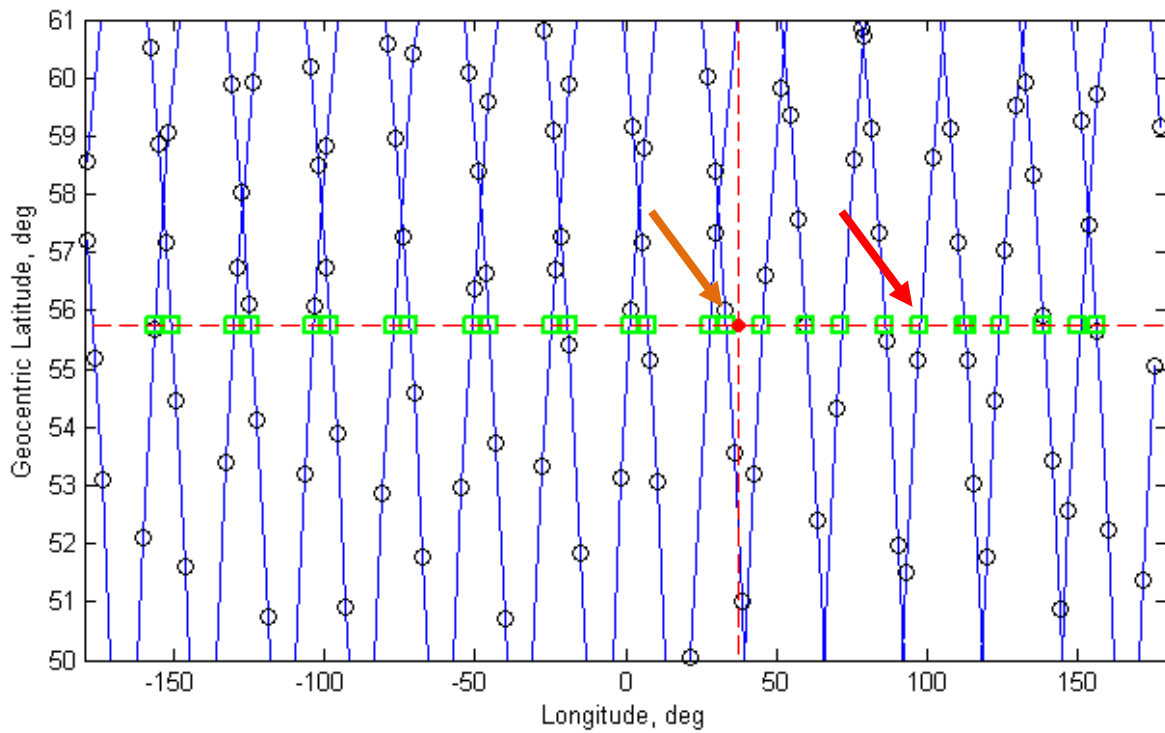


Figure 4.4. Latitude Crossings and Related Longitude Interpolation Solutions
(Trajectory Point: Open Circle; Interpolated Crossing: Square)

As examples, the sample latitude crossings of 97.96 deg E and 33.59 deg E from Fig. 4.4 will be used to create first an “ascending,” then a “descending” phasing maneuver, respectively. For both cases, the type of phasing maneuver is dictated by the location at which the ground track trajectory crosses the line of latitude of the target. With the first example crossing located east of the target, indicated by the right arrow in Fig. 4.4, the TAV traveled too far during the simulation time and overshoot the target. As a result, the semi-major axis of the reference orbit must increase to create an elliptical trajectory defined by a perigee location coinciding with the reference altitude of 1000 km. By conducting a single impulsive, tangential burn the TAV will enter the “ascending” eccentric orbit so as to decrease the angular distance traversed during the orbit period, thus permitting an over-flight of the target rather than a miss to the east.

The amount by which the semi-major axis of the reference orbit must increase is dictated by both the value of $\Delta\theta$ and the number of reference orbits required to produce the elapsed simulation time corresponding to the latitude crossing. The number of reference orbits, n_{ref} , is calculated by dividing the latitude crossing time by the reference orbit period, and then subsequently truncating the result to yield an integer value. Since the Earth rotates at an angular rate of 15 deg per hour, a delta-period, or $\Delta\mathbb{P}$, is calculated by dividing the longitudinal difference, $\Delta\theta$, by the number of reference orbits and then converting into a time duration:

$$\Delta\mathbb{P} = (\Delta\theta/n_{ref})(\text{hr}/15 \text{ deg})(3600 \text{ s/hr}) \quad (4.4)$$

With the latitude crossing located east of the target, the value for $\Delta\mathbb{P}$ must be added to the reference orbit period to produce the “ascending” eccentric, or *perturbed* orbit period. For the east latitude crossing case, the period of the perturbed orbit is 2.118 hr, which corresponds to an “ascending” semi-major axis of 8372.10 km obtained from:

$$a_{pert} = \left(\mu \left(\frac{\mathbb{P}}{2\pi} \right)^2 \right)^{\frac{1}{3}} \quad (4.5)$$

Prior to the propagation of the perturbed orbit, the secant method was again utilized to determine the requisite initial values for heading angle and orbital velocity for the “ascending” maneuver relative to the rotating reference frame. Following the completion of the perturbed orbit propagation – a time equal to the product of the perturbed orbit period and the number of reference orbits – a second impulsive tangential burn is applied when the flight-path angle is $\phi = 0$ deg so as to minimize the ΔV required for orbit re-circularization at the initial reference orbit altitude. Figure 4.5 shows both the propagated perturbed and re-circularized orbits (dashed line) in contrast to the initial reference orbit (solid line).

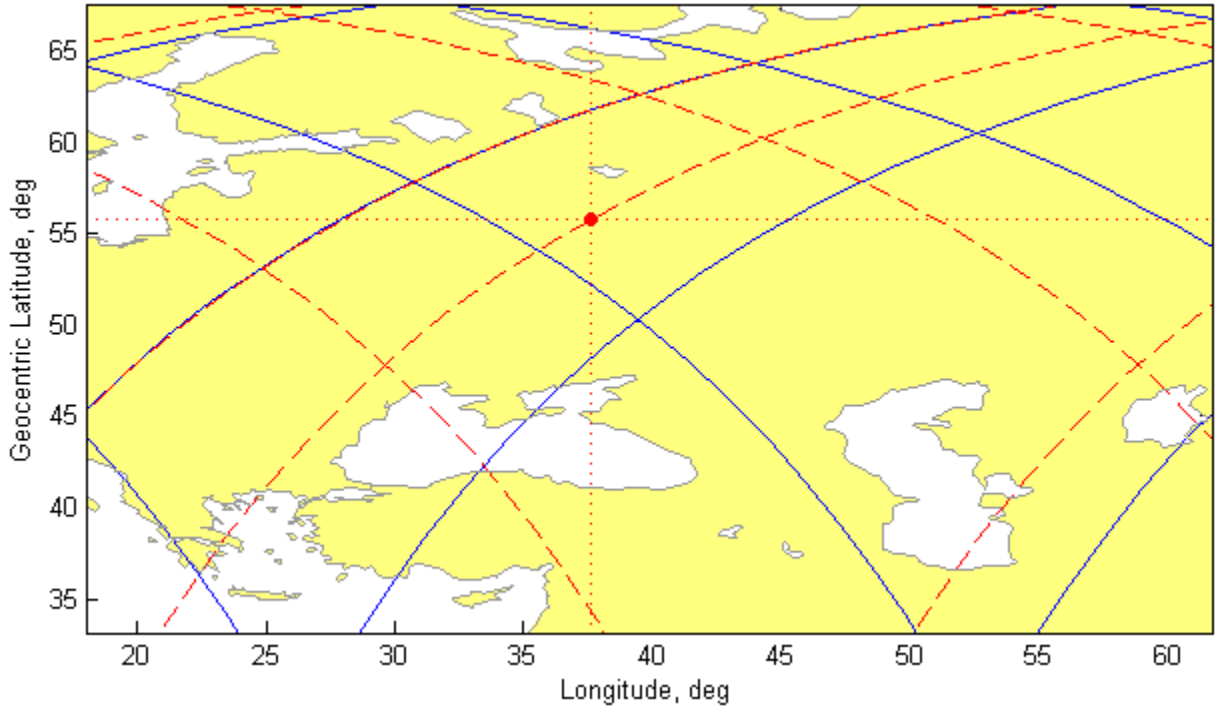


Figure 4.5. Ground Track Trajectory of “Ascending” Phasing Maneuver Example
(Reference Orbit: Solid Line; Perturbed Orbit: Dashed Line)

By conducting an “ascending” phasing maneuver, the TAV shifted the east-latitude crossing westward and produced an ascending-node target over-flight with an elapsed time-of-arrival of 23.59 hr from the simulation start time of $t = 0$ at the initial latitude/longitude coordinates $(\theta, \phi) = (0, 0)$ deg. The total ΔV for the phasing maneuver is 0.846 km/s, comprising of 0.423 km/s for both the de-orbit burn from the reference into the “ascending” eccentric orbit trajectory and orbit re-circularization at the initial reference altitude.

For the “descending” phasing maneuver case, latitude crossings located west of the target indicate that the TAV traveled an insufficient distance during the simulation time and, therefore, undershot the target. Rather than increasing the reference orbit semi-major axis as with the “ascending” case, the “descending” case must instead decrease the semi-major axis to create an elliptical trajectory defined by an apogee location coinciding with the reference altitude of 1000 km. By conducting a single impulsive, tangential burn similar to the “ascending” case, the TAV will enter the “descending” eccentric orbit to overfly the target by traversing a greater angular distance during the orbit period, and thus decreasing the westward longitudinal difference $\Delta\theta$ to zero. Based on the example latitude crossing of 33.59 deg E located west of Moscow, the value for $\Delta\mathbb{P}$ calculated from Eq. (4.4) must be subtracted from the reference orbit period to produce a “descending” eccentric orbit period of 1.685 hr which corresponds to a semi-major axis of 7188.43 km. Figure 4.6 shows the propagated perturbed and re-circularized orbits (dashed line) in contrast to the initial reference orbit (solid line).

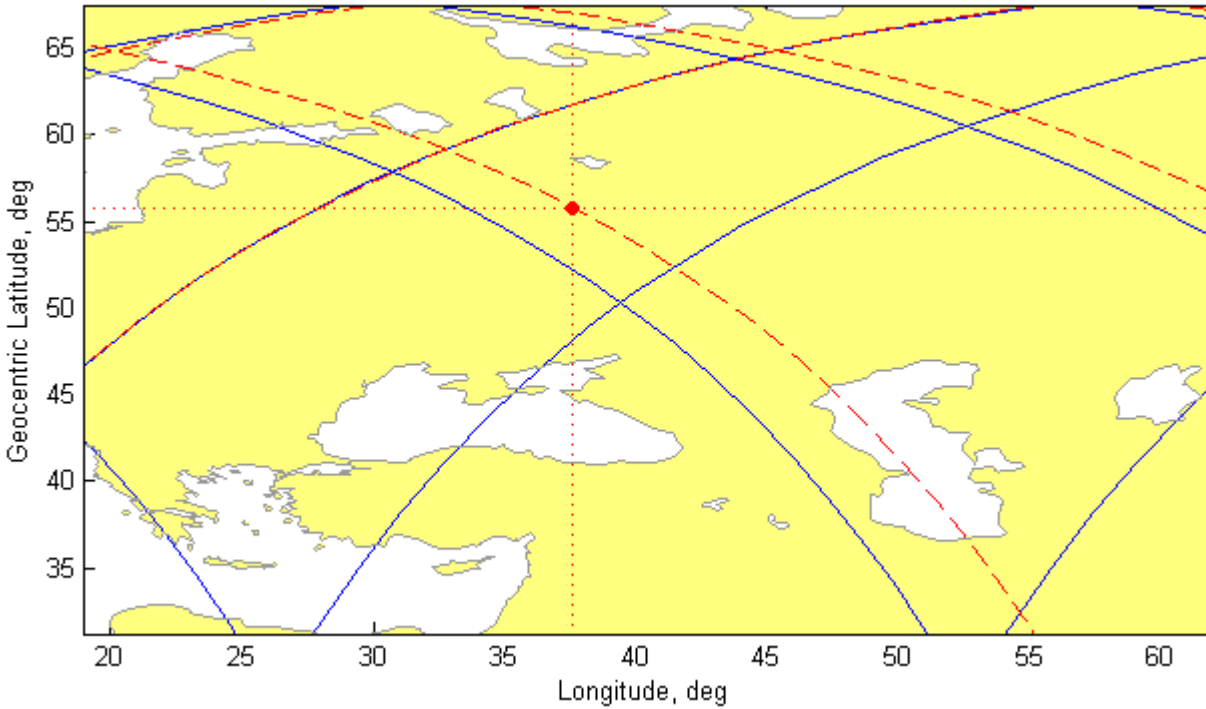


Figure 4.6. Ground Track Trajectory of “Descending” Phasing Maneuver Example
(Reference Orbit: Solid Line; Perturbed Orbit: Dashed Line)

Through the execution of a “descending” phasing maneuver, the TAV shifted the west-latitude crossing eastward and produced a descending-node target over-flight with a time-of-arrival of 7.316 hr. The total ΔV for the phasing maneuver is 0.195 km/s, to include 0.0974 km/s for the tangential burn to transition from the reference to the “descending” eccentric orbit trajectory and 0.0974 km/s for orbit re-circularization.

As an alternative, a modified phasing maneuver is available which transforms “descending” maneuvers into the “ascending” variant. In order to eliminate the longitude difference between a west crossing and the target, “descending” phasing maneuvers reduce the semi-major axis of the reference orbit and thus increase orbital velocity. While theoretically tenable, large values of $\Delta\theta$ generate practical problems since the maneuver semi-major axis produces an impact scenario with the planetary surface. Rather than branding such west crossing

cases as infeasible, they can be transformed into “ascending” phasing maneuvers by subtracting the longitude difference $\Delta\theta$ from 360 deg to create an east crossing on the target line of latitude. As a consequence of this longitudinal shift, the time of the new east crossing is determined by first dividing the modified longitude difference ($360 \text{ deg} - \Delta\theta$) by 360 deg per sidereal day, and then adding the result to the time associated with the original west crossing. Once calculated, the time of the new east crossing is divided by the reference orbit period to yield the requisite number of reference orbits, n_{ref} , to complete the maneuver and overfly the target. Subsequent steps in the maneuver algorithm to include the calculation of ΔP , the perturbed orbit period, and the perturbed orbit semi-major axis, are identical to the conventional “ascending” case.

For all phasing maneuvers analyzed, the time-of-arrival is derived from a determination of miss distance between the ground track trajectory and the target coordinates. Although the preceding discussion indicated that both the “ascending” and “descending” phasing maneuver examples produced a target over-flight, a comparison of the respective ground track trajectories with the target coordinates reveals a distance deviation and, therefore, not a true over-flight despite the target remaining within the field of view of the TAV. Quantitatively, this deviation is expressed by a miss distance of 10.23 km for the “ascending” and 0.68 km for the “descending” example cases.

Ideally, the minimum miss distance between the ground track trajectory and target can be represented as a position vector originating from the target and orthogonally extending to the trajectory. Since the ground track trajectory is comprised of a set of discrete points rather than a continuous curve, the determination of the minimum miss distance can become computationally expensive with the active searching of regions along the trajectory that potentially contain a minimum, then the subsequent interpolation of these candidate regions to provide the points

necessary to calculate the distance between the trajectory and target. As an alternative, the miss distance is determined by first interpolating the coordinates at which the trajectory crossed the lines of latitude and longitude pinpointing the target. Once identified, the distance between these crossings and the target is then calculated with Eq. (4.6) and compared to determine the minimum value.¹¹² See Appendix B for geodesic distance on a non-spherical planetary model.

$$s = r_{\oplus} \cos^{-1}[\sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos(\theta_1 - \theta_2)] \quad (4.6)$$

Simulation of Out-of-Plane Skip Entry Maneuvers

Rather than executing a phasing maneuver, target over-flight can be achieved with out-of-plane skip maneuvers that vary the trajectory perigee altitude and TAV bank angle instead of the reference orbit semi-major axis. By decreasing the perigee altitude below the upper limit of the sensible atmosphere, the TAV encounters increased collisions with atmospheric chemical species as the rarefied, free-molecular flow of the exosphere shifts into the slip-flow, and eventually the hypersonic continuum flow regime of the lower atmospheric layers.¹¹³ With atmospheric density increasing as altitude decreases, the introduction of a non-zero bank angle by the TAV creates an aerodynamic force that enables a change in velocity vector direction and, therefore, the orientation of the orbital plane.

While an optimum out-of-plane solution of minimum target time-of-arrival would involve a simultaneous solution for perigee altitude and bank angle, an alternative method limits the design space and reduces the number of dynamic variables to either: (1) Perigee altitude; or (2) bank angle. If the former option is selected, then the values for both the perigee and skip apogee altitudes are known *a priori*. Since an out-of-plane maneuver is conducted within the

¹¹² Paul Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind, *Geographic Information Systems and Science* (Hoboken, NJ: John Wiley & Sons, Inc., 2005), 117.

¹¹³ King-Hele, 26.

sensible atmosphere, then aerodynamic drag effects produce a skip apogee lower than the reference orbit altitude. The exact value of skip apogee represents an unknown quantity since the magnitude of the bank angle remains to be optimized. By convention, a bank angle approaching ± 90 deg produces an apogee altitude closer to the upper limit of the sensible atmosphere, whereas a bank angle approaching 0 deg yields an apogee altitude closer to that of the reference orbit. Depending on the apogee altitude, the amount of ΔV expended for re-circularization remains variable since the mission might necessitate a boost to a higher altitude if the apogee altitude is too low due to considerations of either payload effectiveness and/or TAV mission lifetime.

With the latter option, only the bank angle is known *a priori* and the perigee altitude remains to be optimized. So as to create an approximate maximum aerodynamic force, the bank angle is set to either ± 90 deg depending on the location of target relative to the ground track trajectory of the reference orbit. Of the two options available, the constant bank angle option was selected and an iterative solution method implemented to optimize perigee altitude for all subsequent analysis. Starting from the reference orbit states given in Table 4.2, the requisite perigee altitude to produce an over-flight condition is determined by first identifying the orientation of the closest approach of the reference orbit ground track trajectory to the target. If south or east, then the bank angle is set to $+90$ deg for a right bank; -90 deg for a left bank if north or west. If the calculated miss distance between the trajectory and target exceeds a specified error tolerance, then the perigee altitude is either decremented to reduce the out-of-plane shift of the trajectory, or, conversely, incremented to increase the trajectory shift. Once over-flight is achieved within permissible miss distance tolerances, the time-of-arrival and total ΔV are then calculated according to the method described for the planar phasing maneuvers.

Simulation of Simple Plane Change Maneuvers

As a means of evaluating the effectiveness of planar phasing and out-of-plane skip maneuvers, the respective time-of-arrival and total ΔV required for ground target over-flight is compared with corresponding values calculated for a simple plane change. Similar to the skip maneuvers, the amount of plane change required for target over-flight is determined by first identifying whether the closest approach of the reference orbit ground trajectory is north or south of the target. If the calculated miss distance between the trajectory and target exceeds a specified error tolerance, then the initial inertial heading angle of the spacecraft is decremented if missing to the north, and, conversely, incremented if missing to the south. Once the heading angle required for target over-flight is obtained, then it is differenced with that of the reference orbit to produce a “delta” value describing the amount of heading angle change required for the maneuver ($\Delta\psi$). A function of relative orbital velocity, flight-path angle, and inclination change, an expression for the ΔV necessary to perform a simple plane change maneuver is given by:¹¹⁴

$$\Delta V_{Simple} = 2 {}^R V_i \cos \gamma \cdot \sin \left(\frac{1}{2} |\Delta i| \right) \quad (4.7)$$

Results and Analysis

The ability for planar phasing and out-of-plane skip maneuvers to perform an over-flight of a specified ground target in minimum time was analyzed for the locations given in Table 4.1. While all sample ground targets were analyzed, planar phasing maneuvers were only implemented for locations deemed representative of the high (Moscow), medium (Gibraltar), and low-latitude (Pontianak) regions. Over-flights of the remaining ground targets were executed utilizing only the skip entry and simple plane change maneuvers.

¹¹⁴ Vallado, 345-346.

Maneuver Performance Comparison for Select Ground Targets

For all sample ground targets, “ascending” and “descending” phasing maneuvers were designed based on the location of the reference orbit relative to the ground target. Employing a fourth-order Runge-Kutta solver, simulations of each phasing maneuver yielded over-flight data featuring not only time-of-arrival and altitude-of-arrival, but also the ΔV required to enter the perturbed orbit and subsequently re-circularize after completion of the required number of reference orbits comprising the maneuver. Based on this data, a series of plots were created to illustrate: (1) ΔV versus time-of-arrival; (2) ground resolution versus time-of-arrival; (3) altitude-of-arrival versus time-of-arrival; and (4) number of reference orbits versus ΔV . Illustrated in Fig. 4.7 for the ground target of Moscow, over-flights originating from a reference orbit altitude of 1000 km occur at an elapsed time of approximately 7.3, 23.6, and 31.3 hr for the planar phasing maneuver cases. For each of these time-of-arrival bands, the solutions corresponding to high values for ΔV indicate that large shifts in latitude are required to create an over-flight. Likewise, the low ΔV solutions arise from small shifts in latitude necessary for target over-flight. In terms of fuel expenditure, the phasing maneuver ΔV decreases as the number of reference orbits increases depending on the semi-major axis of the perturbed orbit. Of the various phasing maneuvers simulated, an “ascending” case with 13 reference orbits and an apogee altitude of 1219.15 km yielded the lowest ΔV at 0.107 km/s. Despite having the same number of reference orbits, an example of an “ascending” case transformed from a “descending” maneuver produced a higher apogee altitude at 4248.40 km and a greater ΔV of 1.268 km/s.

As an alternative initial condition, the reference orbit and related phasing maneuvers were also simulated from an initial altitude of 750 km. While sharing the same time-of-arrival bands as the 1000 km altitude alternative, the 750 km altitude cases produced additional times-

of-arrival at 8.16 and 17.8 hr. For both initial altitude cases, the TAV overflew Moscow at an altitude equal to the initial condition, with the exception of an “ascending” case whose semi-major axis was defined by an apogee and perigee of 12048.20 km and 750 km, respectively, and an altitude-of-arrival of 1385.82 km. To ascertain maneuver effectiveness in terms of altitude-of-arrival, the TAV flies a visible imager payload with dimensions of $(l, w, h) = (2.10, 1.20, 2.80)$ m, an aperture diameter (D) of 1.15 m, a focal length (f) of 2.70 m, and image wavelength of 1.0 μm . Using Eq. (4.8), the diffraction-limited ground resolution for each maneuver is calculated with respect to the altitude-of-arrival over each sample ground target.¹¹⁵

$$X_{vis} = 2.44 h \lambda D^{-1} \quad (4.8)$$

For the 1000 km initial altitude case, the ground resolution was 2.12 m, while the resolution decreased to 1.59 m for the 750 km case.

In addition to planar phasing maneuvers, Fig. 4.7 also shows the over-flight parameters for two out-of-plane skip maneuvers performed from an initial altitude of 1000 km. For an initial inclination of 70 deg, only a single out-of-plane maneuver opportunity is available for the target latitude crossing at 33.59 deg E. Banking at $\sigma = -90$ deg, this maneuver produced an over-flight time-of-arrival of 7.361 hr with $\Delta V = 0.482$ km/s. When the inertial inclination is decreased to 60 deg, however, two out-of-plane maneuver opportunities become available. As outlined in Table 4.3, the skip maneuvers surpassed the majority of phasing maneuvers in terms of the ΔV required to achieve the shortest time-of-arrival. Although a “descending” phasing maneuver was shown to overfly Moscow in 7.316 hr with $\Delta V = 0.195$ km/s, the first skip maneuver with $\sigma = +90$ deg and an initial inclination of 60 deg was able to achieve an over-

¹¹⁵ Bruce Chesley, Reinhold Lutz, and Robert F. Brodsky, “Space Payload Design and Sizing,” in *Space Mission Analysis and Design*, ed. James R. Wertz and Wiley J. Larson (El Segundo, CA: Microcosm Press, 2003), 264.

flight after 1.947 hr with $\Delta V = 0.466$ km/s – a ΔV increase of 139% for a time-of-arrival savings of 5.369 hr. Likewise, an increase in ΔV from 0.466 km/s to 0.485 km/s produces an over-flight after 5.611 hr for the second 60 deg inclination case – a time-of-arrival that is 1.705 hr faster than the preceding “descending” phasing maneuver case. Besides faster times-of-arrival over Moscow, the skip maneuver examples furthermore produce improved imager resolutions of 1.86 m and 2.10 m, respectively, since the re-circularized orbit altitudes of 876.57 km and 989.97 km are lower than the reference orbit altitude.

As a final point of comparison, Fig. 4.7 also presents the time-of-arrival and ΔV required to produce an over-flight of Moscow via a simple plane change maneuver. At an altitude of 1000 km, the simple plane change achieves a time-of-arrival of 2.043 hr with $\Delta V = 0.491$ km/s. By comparison, the $\sigma = +90$ deg skip entry case produced a 0.096 hr-slower time-of-arrival with $\Delta V = 0.516$ km/s, thus making the simple plane change the superior alternative. In terms of time-of-arrival alone, the simple plane change out-performs the “ascending” and “descending” phasing maneuvers alike, while for ΔV , it under-performs the “ascending” maneuver with $\Delta V = 0.107$ km/s. When the altitude is decreased to 750 km, then the simple plane change achieves an over-flight of Moscow with $\Delta V = 0.0128$ km/s. While this represents the lowest ΔV value among the various maneuver options, the lower altitude produces a trade-off with a time-of-arrival of 23.576 hr for the simple plane change.

Table 4.3. Out-of-Plane Skip Maneuver Parameters for Moscow, Russia

<i>Parameter</i>	$i_i = 70$ deg	$i_i = 60$ deg	$i_i = 60$ deg
Bank Angle	−90 deg	+90 deg	−90 deg
Latitude Crossing	33.59 deg E	26.36 deg E	35.06 deg E
Time-of-Arrival, hr	7.361	1.947	5.611
ΔV_{Total} , km/s	0.482	0.466	0.485
h_p , km	95.9	88.39	103.1
Altitude-of-Arrival, km	949.01	876.57	989.97
X_{vis} , m	2.01	1.86	2.10
Miss Distance, km	0.637	1.85	0.047

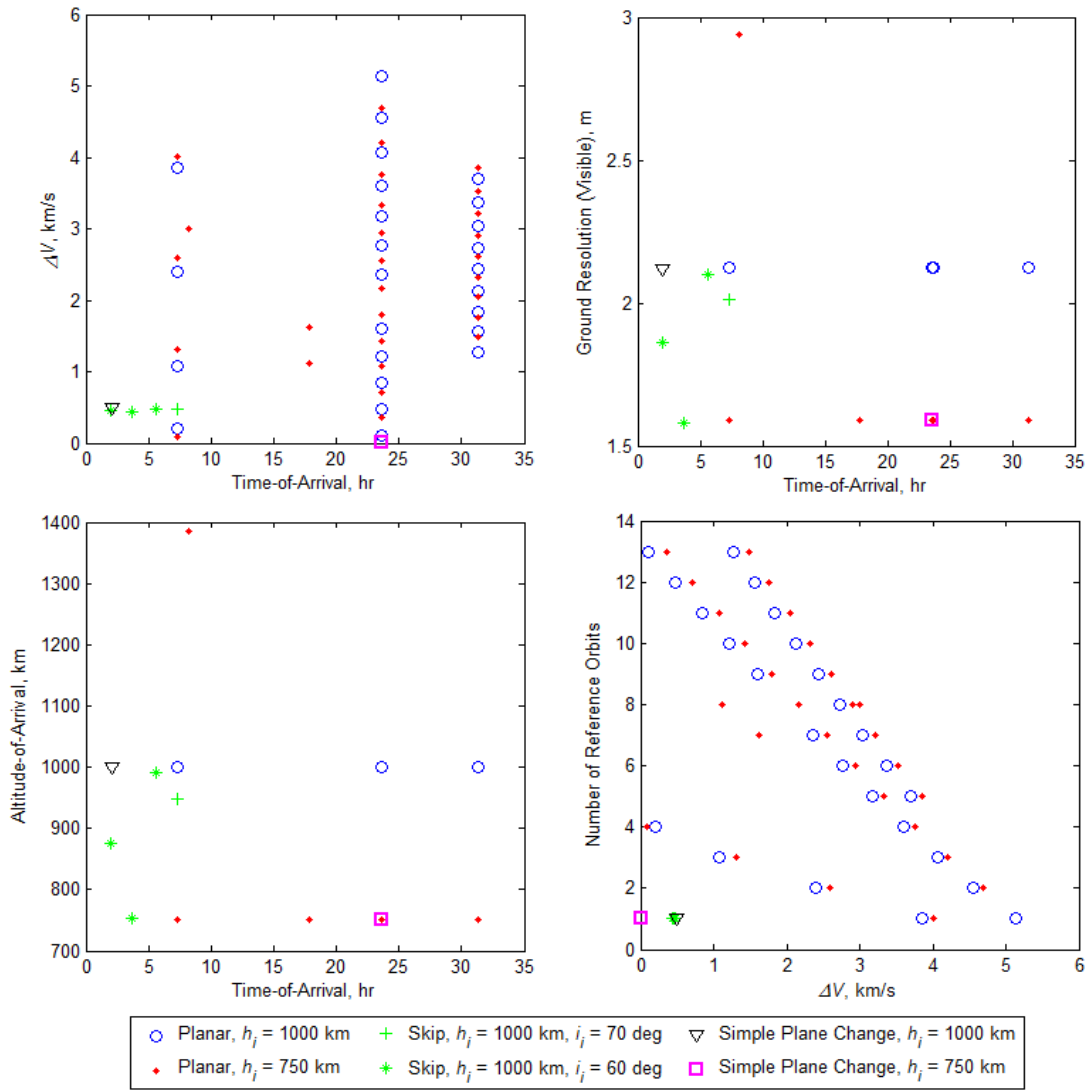


Figure 4.7. Maneuver Over-Flight Parameters for Moscow, Russia

For the medium-latitude case of Gibraltar, over-flights originating from an initial altitude of 1000 km also occur in bands, but at elapsed times of approximately 11.3, 25.3, and 35.2 hr for the phasing maneuver cases. As shown in Fig. 4.8, the phasing maneuvers commencing at an altitude of 1000 km maintained a lower ΔV than the 750 km-case within the 11.3 hr time-of-arrival band. Similar to the Moscow case, phasing maneuvers commencing at 1000 km outperformed the simple plane change maneuver at the same initial altitude with $\Delta V = 0.046$ km/s – a value 0.179 km/s lower than the simple plane change with $\Delta V = 0.225$ km/s and a time-of-arrival of 11.19 hr. Although more expensive in terms of ΔV , the simple plane change maneuver conducted at a 750 km altitude produced the fastest time-of-arrival at 1.86 hr.

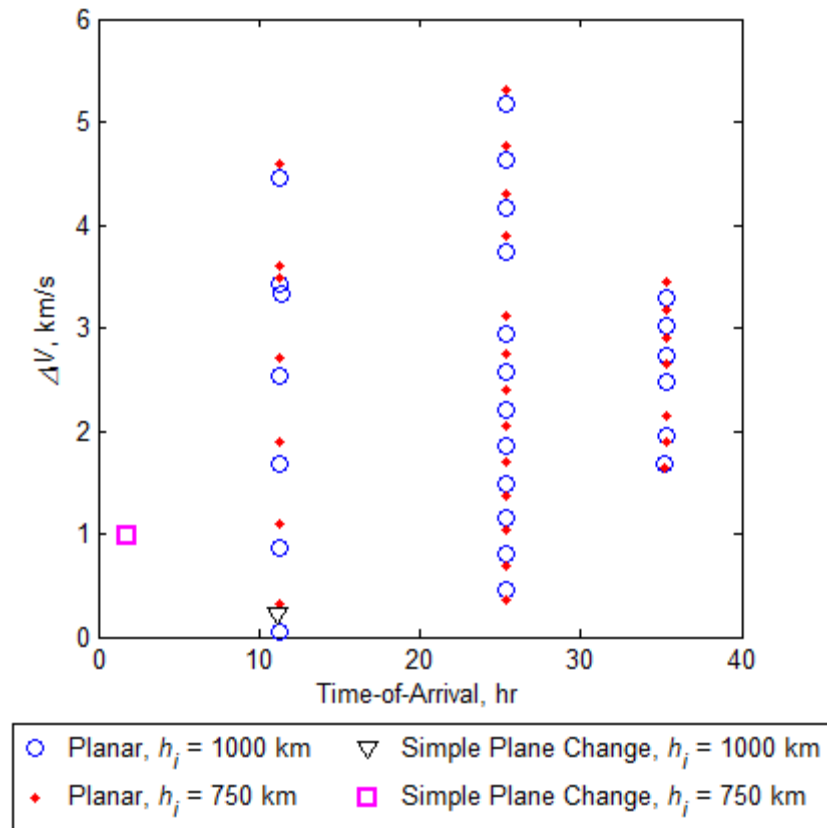


Figure 4.8. Maneuver Over-Flight Parameters for Gibraltar, United Kingdom

Lastly, for the equatorial case of Pontianak, Indonesia, over-flights originating from an initial altitude of 1000 km occur at elapsed times of approximately 4.7, 16.6, 28.6, and 40.6 hr for the phasing maneuver cases. As compared with Moscow and Gibraltar, the 40.6 hr band represents the longest for time-of-arrival and results from the transformation of “descending” phasing maneuvers into the “ascending” alternative. Depicted in Fig. 4.9, the phasing maneuvers commencing at both the 750 km and 1000 km initial altitude cases maintained a considerably lower ΔV than the simple plane change maneuvers, with the most expensive phasing maneuver at a ΔV of 4.293 km/s, a value 45.1% lower than $\Delta V = 7.815$ km/s for the 1000 km-altitude simple plane change. Overall, such disparity in ΔV between the phasing and simple plane change maneuvers stems from the mechanics of the maneuvers: the former achieves over-flight by either increasing or decreasing the reference orbit semi-major axis while retaining the original heading angle and inclination; the latter achieves over-flight by decreasing the inclination angle from 70 deg to 0 deg. As a consequence of its equatorial location, Table 4.4 shows that Pontianak requires the highest ΔV among the various sample ground targets to achieve over-flight via simple plane change. For the remaining locations, a direct relationship between ΔV and target latitude cannot be conclusively established since the values listed reflect the ΔV required to shift the reference orbit ground track towards the target with the intent of creating an over-flight.

Table 4.4. Simple Plane Change Maneuver Parameters ($h_i = 1000$ km, $i_i = 70$ deg)

<i>Ground Target</i>	$\psi_{Simple}, \text{deg}$	<i>Time-of-Arrival, hr</i>	$\Delta V_{Simple}, \text{km/s}$
Reykjavik, Iceland	66.65	9.254	0.418
Moscow, Russia	73.25	2.043	0.491
Tokyo, Japan	64.65	15.96	0.666
Gibraltar, United Kingdom	68.20	11.19	0.225
Pontianak, Indonesia	0.000	4.355	7.815

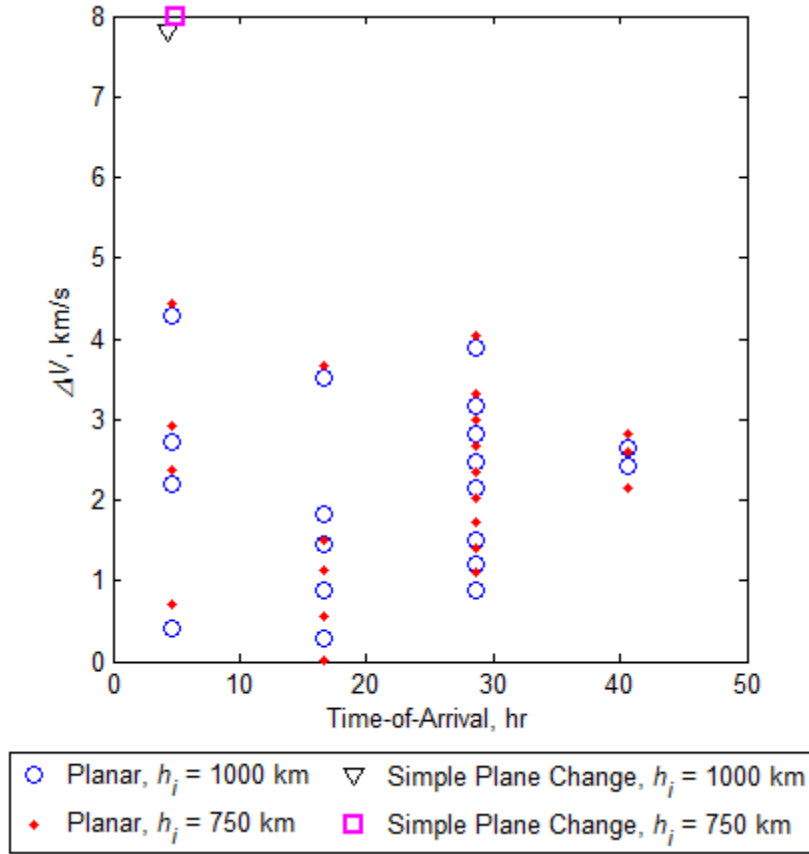


Figure 4.9. Maneuver Over-Flight Parameters for Pontianak, Indonesia

Analysis of Out-of-Plane Skip Entry Maneuvers

For the Moscow over-flight scenario, the first skip maneuver executed set the TAV bank angle to $\sigma = +90$ deg and the perigee altitude at 88.39 km so as the shift the target latitude crossing at 26.36 deg E eastward to overfly the target. By performing a rightward-bank, the skip maneuver not only shifted the ground track trajectory of the reference orbit to the south and east, but also decreased the maximum orbit inclination from 60 deg to 57.95 deg, a reduction of 3.42%. Even though $\sigma = +90$ deg at the start of the simulation, a shifting in the perturbed orbit with respect to the reference orbit does not occur until the altitude of the TAV approaches the upper limit of the sensible atmosphere and descends below it.

Representing an ascending node over-flight opportunity, the ground track trajectory of the first skip maneuver is shown in detail in Fig. 4.10. As a result of aerodynamic drag encountered by the TAV near perigee, the skip apogee altitude and resulting re-circularized orbit altitude of 876.57 km is 12.34% lower than the initial 1000 km altitude. Re-circularized at a comparatively high altitude low-Earth orbit, the TAV is capable of performing either subsequent exo- or trans-atmospheric maneuvers due to the higher orbital potential energy available.

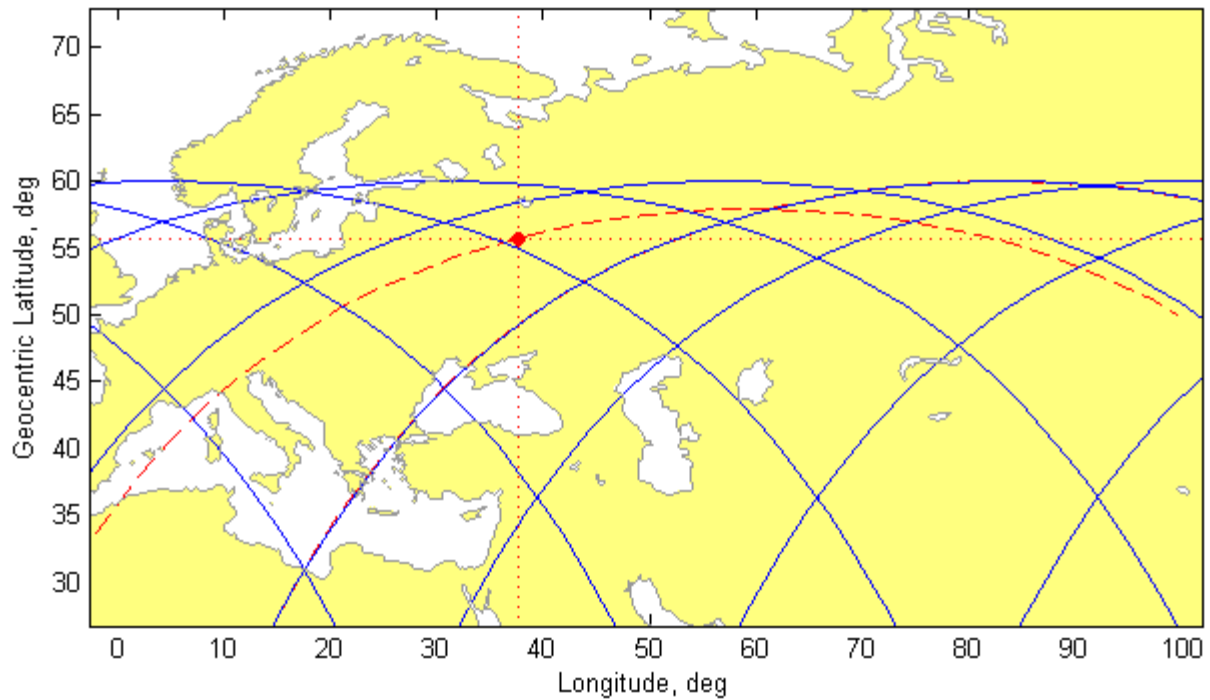


Figure 4.10. Over-Flight Detail of Ascending Node Out-of-Plane Skip Maneuver
(Reference Orbit: Solid Line: Perturbed Orbit: Dashed Line)

For the second skip maneuver, the TAV bank angle was set to $\sigma = -90$ deg and perigee optimized at a higher altitude of 103.1 km so as to shift the target latitude crossing at 35.06 deg E eastward towards the target. By performing a maximum bank to the left as opposed to the right as in the first out-of-plane case, the skip maneuver in Fig. 4.11 shifted the ground track trajectory of the reference orbit to the north and east, thereby decreasing the maximum orbit inclination from 60 deg to 59.82 deg, a reduction of 0.3%. Since the skip entry seeks to

shift a descending node segment of the reference orbit, more flight time is available to propagate the change in the orbital plane created by the skip maneuver. As a result, the perigee altitude is optimized at a higher altitude of 103.1 km in order to reduce the aerodynamic drag encountered by the TAV and limit the overall change in orbit inclination. When simulated, the second skip maneuver case re-circularized at an orbit altitude of 989.97 km, a reduction of 1.00% from the initial 1000 km altitude.

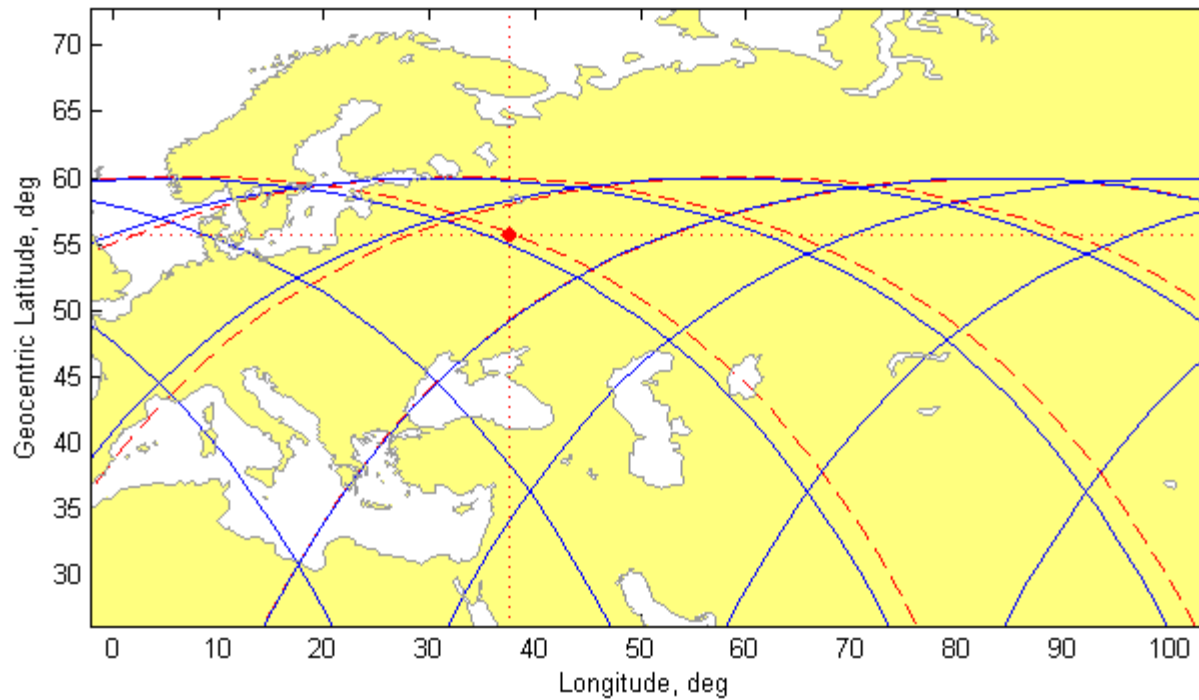


Figure 4.11. Over-Flight Detail of Descending Node Out-of-Plane Skip Maneuver
(Reference Orbit: Solid Line: Perturbed Orbit: Dashed Line)

Similar to the preceding maneuver case, the skip entry performed at an initial inclination of 70 deg also maintained a bank angle of $\sigma = -90$ deg. Transiting a perigee altitude of 95.9 km, this maneuver achieved a time-of-arrival of 7.361 hr with $\Delta V = 0.482$ km/s. While out-performed by phasing maneuvers executed at the same initial conditions, the skip maneuver at $i_i = 70$ deg still provides a responsive over-flight trajectory with a time-of-arrival less than 18 hr and $\Delta V = 0.5$ km/s.

Maneuver Performance Comparison for All Ground Targets

For the complete list of sample ground targets, encompassing both northern and southern hemisphere locations, only the skip entry and simple plane change maneuvers were simulated to determine the relative performance of each out-of-plane maneuver option. Starting from the latitude/longitude coordinates $(\theta, \phi) = (0, 0)$ deg, a simulation time of $t = 0$, and an initial inclination of 60 deg, Table 4.5 illustrates the target time-of-arrival, Δt , and ΔV required to achieve target over-flight. Since the simple plane change maneuvers occur *in vacuo*, then metrics related to maximum deceleration and maximum stagnation heat flux are provided only for the skip maneuver alternative. While only one simple plane change opportunity is available for each target, upwards of one to possibly four maneuver opportunities are available for skip entry based on the placement of the reference orbit ground track vis-à-vis the target. For example, two skip maneuver opportunities exist for Tokyo, while four opportunities exist for Cape Town.

Upon comparison, the skip maneuvers produced the lowest mean ΔV , with the required ΔV expenditure for each target and associated set of maneuver opportunities being less than 0.5 km/s. Shown in Fig. 4.9 and Table 4.4, the simple plane change produced the highest ΔV expenditure for the over-flight of Pontianak at 7.096 km/s, compared with $\Delta V = 0.443$ km/s for skip entry. Although the simple plane change maneuver demonstrated a faster time-of-arrival for the targets of Buenos Aires, Brasilia, Canberra, Pontianak, Reykjavik, and Tokyo, the skip maneuvers out-performed each of these cases in terms of ΔV . Of the sample targets selected, Gibraltar and Moscow represent the only over-flight cases in which the ΔV expenditure for the simple plane change out-performed that of the skip maneuver.

Table 4.5. Skip Entry and Simple Plane Change Maneuver Comparison
($h_i = 1000$ km, $i_i = 60$ deg)

<i>Ground Target</i>	<i>Simple Plane Change</i>			<i>Skip Entry</i>				
	Δi , deg	<i>Time-of-Arrival</i> , hr	ΔV , km/s	Δi , deg	<i>Time-of-Arrival</i> , hr	ΔV , km/s	<i>Max. Decel.</i> , g	$\dot{Q}_{s,max}$, kW/m ²
Reykjavik	4.13	7.446	0.511	4.61	8.224	0.364	0.278	497.32
Moscow	0.74	5.785	0.092	0.05	1.947	0.466	0.135	344.87
				0.05	3.652	0.434	0.267	487.84
				0.07	5.611	0.485	0.131	82.73
Tokyo	4.67	15.963	0.578	4.14	15.991	0.378	0.253	474.53
				0.05	16.609	0.425	0.304	519.92
Gibraltar	0.78	1.959	0.097	4.11	1.731	0.379	0.252	473.64
				0.05	9.532	0.311	0.572	708.11
Pontianak	60.00	4.355	7.096	0.05	16.644	0.443	0.230	452.13
Brasilia	24.78	1.615	3.045	10.64	15.433	0.304	0.375	577.43
Buenos Aires	10.09	1.519	1.248	8.97	2.837	0.299	0.369	572.91
				0.05	18.076	0.355	0.476	649.00
Canberra	4.97	12.046	0.615	3.48	12.636	0.396	0.219	441.23
				0.05	10.921	0.392	0.391	589.73
Cape Town	5.29	20.813	0.655	5.17	21.501	0.350	0.302	518.84
				4.17	12.004	0.376	0.256	477.52
				0.05	20.677	0.397	0.379	580.70
				0.05	12.543	0.432	0.277	496.63

Increasing the initial inclination of the reference orbit from 60 deg to 70 deg produced similar results to those given in Table 4.5, with the skip maneuvers maintaining a mean ΔV less than 0.5 km/s for each target over-flight, as well as a maximum deceleration and stagnation heat flux less than 1.0 g and 1000 kW/m², respectively. Although the simple plane change maneuver provided a faster time-of-arrival than skip entry for several targets in both Tables 4.5 and 4.6, to include Brasilia and Pontianak, the ΔV expenditure is considerably greater. For example, an over-flight of Buenos Aires commencing from a $i_i = 60$ deg reference orbit achieves a time-of-arrival of 1.519 hr for a simple plane change, compared with 2.837 hr for the fast skip entry opportunity. Despite saving 1.318 hr in flight time, the simple plane change requires $\Delta V =$

1.248 km/s, a 317% increase from the ΔV required for the skip maneuver. Similarly, an over-flight of Canberra commencing from a $i_i = 70$ deg reference orbit achieves a time-of-arrival of 2.807 hr for a simple plane change, while the fastest skip entry opportunity achieves over-flight in 12.546 hr. Despite producing a faster time-of-arrival of 9.739 hr, the simple plane change requires a 77% greater ΔV expenditure than the skip entry alternative.

Even though values for maximum deceleration and stagnation heat flux are presented in Tables 4.5 and 4.6, the relative impact of these parameters as maneuver performance measures only attain significance when compared with existing re-entry deceleration and heat flux data. When trajectory data for vehicles such as the Apollo Command Module or Space Shuttle is examined, it becomes apparent that the simulated deceleration and heat flux experienced by the TAV are considerably lower in magnitude, with deviations primarily arising due to the perigee altitude selected for the skip trajectory. With Apollo and the Space Shuttle performing a terminal re-entry rather than a skip entry aeroassisted maneuver, the vehicles experience an exponentially increasing dense atmosphere as the altitude decreases towards sea-level. Consequently, increased atmospheric density translates into greater deceleration and heat flux experienced by the vehicle as kinetic energy decreases and is frictionally converted into heat during re-entry.

In terms of TAV survivability during the skip maneuver, the maximum deceleration of 0.304 g for Gibraltar from Table 4.5 is favorable since it is less than 1.0 g and one order of magnitude less than the maximum deceleration experienced by vehicles such as Apollo. For example, re-entry of the Apollo 10 Command Module from lunar transfer orbit produced a maximum deceleration of approximately 6.75 g.¹¹⁶ As for stagnation heat flux, TAV survivability is not explicitly evident and thus a comparison with known re-entry data is

¹¹⁶ Hicks, 411.

required. Recorded at an approximate altitude of 70 km, STS-5 experienced a maximum heat flux of 1400 kW/m^2 on the lower surface of the wing leading-edge.¹¹⁷ Despite being lower in magnitude, the maximum skip entry value of $\dot{Q}_{s,max} = 708.11 \text{ kW/m}^2$ from Table 4.5 (also for Gibraltar) only represents an estimate of stagnation heat flux, whereas the STS-5 measurement is total heat flux, to include contributions by radiative heating. Based on the comparatively shallower entry of the TAV, however, the total heat flux is assumed to be less than the maximum STS-5 measurement and is deemed survivable for the notional TAV.

Table 4.6. Skip Entry and Simple Plane Change Maneuver Comparison
($h_i = 1000 \text{ km}, i_i = 70 \text{ deg}$)

Ground Target	Simple Plane Change			Skip Entry				
	$\Delta i, \text{ deg}$	Time-of-Arrival, hr	$\Delta V, \text{ km/s}$	$\Delta i, \text{ deg}$	Time-of-Arrival, hr	$\Delta V, \text{ km/s}$	Max. Decel., g	$\dot{Q}_{s,max}, \text{ kW/m}^2$
Reykjavik	3.29	9.254	0.412	7.49	3.285	0.305	0.376	575.69
				0.03	6.862	0.419	0.363	564.67
Moscow	2.16	7.591	0.271	0.41	7.361	0.482	0.109	166.40
Tokyo	5.31	15.963	0.666	6.05	15.335	0.328	0.356	559.45
				8.44	2.107	0.297	0.387	583.99
				0.03	16.084	0.388	0.444	626.81
				0.03	15.681	0.486	0.110	206.77
Gibraltar	1.74	11.196	0.218	0.03	1.743	0.408	0.392	506.03
				4.56	11.552	0.365	0.295	587.67
Pontianak	70.00	4.355	8.241	0.03	16.650	0.448	0.256	469.08
Brasilia	34.78	1.615	4.294	0.03	15.640	0.426	0.344	548.67
Buenos Aires	0.15	16.825	0.019	2.55	3.042	0.424	0.183	388.82
				0.03	17.151	0.409	0.389	585.17
Canberra	4.47	2.807	0.560	6.52	13.384	0.316	0.363	564.60
				0.03	12.546	0.390	0.440	624.07
Cape Town	2.41	11.562	0.302	7.20	22.138	0.303	0.376	575.32
				7.04	11.347	0.308	0.372	572.12
				0.03	21.563	0.450	0.241	453.08
				0.03	12.091	0.388	0.444	626.52

¹¹⁷ Ko, "Finite Element," 16, 18, 32.

Summary and Conclusion

Based on a notional trans-atmospheric, lifting re-entry vehicle design with $L/D = 6$, a series of planar phasing, out-of-plane skip entry, and simple plane change maneuvers were simulated to overfly a set of sample ground targets located at high-, medium-, and low-latitudes, in the northern and southern hemispheres. From these simulations the creation of time-of-arrival bands was shown, each comprised of a family of phasing maneuver solutions with a corresponding total ΔV dependent on both the type and number of maneuvers performed. Whether characterized as “ascending” or “descending,” phasing maneuvers maintain consistently low ΔV requirements of less than 0.5 km/s, with times-of-arrival less than 18 hr for a variety of ground targets, both east and west of the Prime Meridian. While the ΔV for the simple plane change is lower than most phasing maneuvers executed for over-flights of Moscow and Gibraltar, the equatorial target of Pontianak, Indonesia illustrated that the choice of ground target can have a detrimental impact on ΔV with values approaching 8.0 km/s for a single simple plane change. For a limited sample ground target set, the skip entry aeroassisted maneuvers have been shown to consistently provide responsive mission execution in terms of target time-of-arrival, with maximum deceleration and stagnation heat flux less than 1.0 g and 1000 kW/m², respectively.

V. Design of Experiments Approach to Atmospheric Skip Entry Maneuver Optimization

Chapter Overview

An optimal trans-atmospheric vehicle and trajectory design are presented to simultaneously maximize the change in inclination angle and minimize total ΔV for an atmospheric skip entry maneuver. Utilizing a Design of Experiments approach featuring orthogonal arrays of experiments, the optimal vehicle and trajectory designs are determined within the context of main effects and Pareto front analysis by evaluating the relative performance of six design variables, to include mass, planform area, aerodynamic coefficients, perigee altitude, and bank angle. Depending on the chosen re-circularization altitude, the optimal design performing a skip entry aeroassisted maneuver can achieve an inclination change of 19.91 deg with 50-85% less ΔV than a simple plane change.

Introduction

For the skip entry type of aeroassisted maneuver, maneuver design represents a multi-objective optimization problem (MOP) with a decision space containing not only TAV and trajectory design parameters, but also constraints related to TAV capability, such as available ΔV , maximum deceleration g -loading, and maximum heat flux. With the MOP assumed to be unconstrained in terms of TAV capability, the decision space then focuses on optimizing only the TAV and trajectory designs in order solve the primary MOP defined by:

$$\begin{aligned} \text{MOP} &= \begin{cases} \max_{f(\vec{x})} \Delta i \\ \min_{f(\vec{x})} \Delta V \end{cases} \\ \text{subject to } \vec{x} &\in [m, S, C_D, C_L, h_p, h_i, \sigma] \end{aligned} \quad (5.1)$$

As a sample scenario in which to solve the MOP, a TAV – launched from Wallops Island, VA into a circular orbit with an inclination equal to the launch site latitude (37.84 deg N) – is to perform a skip entry maneuver at a bank angle of $\sigma < 0$ deg. Since the initial reference orbit is prograde, then a negative bank angle produces a leftward turn and, therefore, an increase in orbit inclination angle. Conversely, a positive bank angle creates a rightward turn and a negative change in inclination. Furthermore, the scenario neither requires ground target over-flights at specified times, nor adheres to imposed *no-fly zones* when conducting the maneuver.

While all simulations conducted within the present research perform a single skip entry maneuver, the user of a given TAV maintains the prerogative of performing as many exo- or trans-atmospheric maneuvers as permitted by the ΔV capacity of the vehicle. Consequently, the ability to perform consecutive maneuvers is contingent on the orbital energy of the TAV. With re-circularization required for continued mission operations, the altitude of re-circularization becomes important since the ΔV necessary for orbit injection decreases as the altitude of desired re-circularization increases. Presented as a secondary MOP, the corollary objective space of re-circularization altitude (h_{recirc}) vs. Δi is given in Eq. (5.2). As a tertiary MOP, Eq. (5.3) illustrates the objective space of h_{recirc} vs. ΔV .

$$\begin{aligned} \text{MOP} &= \begin{cases} \max_{f(\vec{x})} \Delta i \\ \max_{f(\vec{x})} h_{recirc} \end{cases} \\ \text{subject to } \vec{x} &\in [m, S, C_D, C_L, h_p, h_i, \sigma] \end{aligned} \quad (5.2)$$

$$\begin{aligned} \text{MOP} &= \begin{cases} \min_{f(\vec{x})} \Delta V \\ \max_{f(\vec{x})} h_{recirc} \end{cases} \\ \text{subject to } \vec{x} &\in [m, S, C_D, C_L, h_p, h_i, \sigma] \end{aligned} \quad (5.3)$$

For each MOP, re-circularization is assumed to occur following the trans-atmospheric flight segment at skip apogee.

Methods of Maneuver Optimization

Whether exo- or trans-atmospheric in nature, maneuver optimization seeks to maximize the ability of a spacecraft to change orbital states while managing constraints linked to propellant availability, mission time factors, and trajectory design. Saddled with limited computing resources, early research into trans-atmospheric maneuver optimization sought to simplify the problem by linearizing the system dynamics as well as introducing dimensionless state variables into the differential equations of motion.¹¹⁸ Once simplified, a classical optimization approach was applied to produce optimal solutions by evaluating expressions for the variational Hamiltonian, Lagrange multipliers, adjoint variables, and terminal transversality conditions. Following this general method, several Mayer-style performance indices were solved for skip entry, to include: (1) maximizing V_f with h_f prescribed, and vice versa for a single maneuver;¹¹⁹ (2) maximizing the orbit inclination change, Δi , for a vehicle conducting a transfer from high Earth orbit to LEO via aerobraking;¹²⁰ (3) maximize inclination change and range for multiple-skip maneuvers;¹²¹ and (4) simultaneously minimize ΔV and maximize skip entry time-of-flight (TOF) while minimizing peak heat flux.¹²²

Modern advancements in computing have enabled the formulation of increasingly robust numerical algorithms which support multiple degrees of freedom trajectory simulations and produce optimal solutions without system linearization or equation non-dimensionalization.

¹¹⁸ Dean R. Chapman, "An Approximate Analytical Method for Studying Entry into Planetary Atmospheres," *NACA TN 4276* (Moffett Field, CA: AMES Aeronautical Laboratory, 1958), 1-101; Eggers and Wong, 1364-1375; J. L. Speyer and M. E. Womble, "Approximate Optimal Atmospheric Entry Trajectories," *Journal of Spacecraft and Rockets* 8 (1971): 1120-1125.

¹¹⁹ N. X. Vinh, A. Busemann, and R. D. Culp, "Optimum Three-Dimensional Atmospheric Entry," *Acta Astronautica* 2 (1975): 593-611.

¹²⁰ N. X. Vinh and John M. Hanson, "Optimal Aeroassisted Return from High Earth Orbit with Plane Change," *Acta Astronautica*, 12 (1985): 11-25.

¹²¹ N. X. Vinh and Der-Ming Ma, "Optimal Multiple-Pass Aeroassisted Plane Change," *Acta Astronautica* 21 (1990): 749-758; N. X. Vinh and Ya-Wen Shih, "Optimum Multiple-Skip Trajectories," *Acta Astronautica* 41 (1997): 103-112.

¹²² Miele et al., 99-122.

Identified as a class of direction collocation, pseudospectral methods parameterize the state and control trajectories and path constraints using interpolating polynomials, thereby converting an optimal control problem into a nonlinear programming problem. When the polynomials are obtained from a Gaussian quadrature, then the method is identified as a Gaussian pseudospectral method.¹²³ A subtype of algorithms which numerically calculate the value of a definite integral in one or more dimensions, Gaussian quadrature utilizes polynomial approximations of the integrand f of increasing degree. The roots of the polynomials, also referred to as nodes, are then chosen optimally to “maximize the degree of polynomials that the quadrature integrates exactly.”¹²⁴ As examples of pseudospectral method implementation, Sun and Zhang maximized the range of a single skip maneuver subject to several path constraints to include g -loading, dynamic pressure, and heat flux,¹²⁵ while Rao *et al.*¹²⁶ and Darby and Rao¹²⁷ minimized ΔV for multiple-skip maneuvers subject to only heat flux path constraints.

In addition to the implementation of numerical algorithms such as pseudospectral methods to solve optimal control problems, computing advances have also enabled the increase in problem complexity with the development of multidisciplinary design optimization (MDO) and metaheuristic methods to solve multistate, multi-objective problems (MOPs).¹²⁸ One method of solving a MOP, and the focus of the present research, utilizes the Design of Experiments (DOE) method of orthogonal arrays to provide optimal solutions based on the simulation of

¹²³ Yong Sun and Maorui Zhang, “Optimal Re-Entry Range Trajectory of Hypersonic Vehicle by Gauss Pseudospectral Method” (Paper presented at the *2nd International Conference on Intelligent Control and Information Processing*, Harbin, China, 25-28 July 2011): 545-549.

¹²⁴ Narayan Kovvali, *Theory and Applications of Gaussian Quadrature Methods* (New York: Morgan & Claypool Publishers, 2011), 2.

¹²⁵ Sun and Zhang, 545-549.

¹²⁶ Rao *et al.*, “Numerical Optimization Study,” 215-238.

¹²⁷ Darby and Rao, “Optimal Impulsive,” 39-52.

¹²⁸ El-Ghazali Talbi, *Metaheuristics: From Design to Implementation* (Hoboken, NY: John Wiley & Sons, Inc., 2009), 308.

optimal control design experiments formulated from a user-defined design space. Besides orthogonal arrays, other statistical techniques exist within the DOE framework to characterize objective space behavior (output) with respect to the points comprising the design space (inputs), to include full-factorial design and Latin-hypercube spacing.¹²⁹ The most computationally intensive, full-factorial design evaluates every combination of design variable, or factor, at every design variable value, or level. As the number of factors and levels increase for a given MOP, the number of experiments within a full-factorial design increases exponentially.¹³⁰ Requiring fewer design experiments than either the full-factorial or orthogonal array alternatives, Latin-hypercube spacing seeks to maximize design space coverage by not only maximizing the distance between design points, but also preserving near-uniform spacing between the points.¹³¹

Although requiring more design experiments than Latin-hypercube spacing, orthogonal arrays permit the calculation of main effects for each factor, which represents the effect of a given factor averaged across all levels of the remaining factors.¹³² Similar to the other statistical techniques, the objective space resulting from the orthogonal array experiment simulations allow for the determination of optimal solutions based on Pareto front analysis and the identification of non-dominated design solutions.¹³³ Apart from disciplines such as biology and chemical engineering,¹³⁴ DOE methods – specifically orthogonal arrays – have been utilized in various aerospace optimization applications to include multi-layer insulation design for re-entry

¹²⁹ Jeremy S. Agte, “Multistate Analysis and Design: Case Studies in Aerospace Design and Long Endurance Systems” (Ph.D. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology (MIT), 2011), 103.

¹³⁰ Roger P. Peterson, *Design and Analysis of Experiments* (New York, NY: Marcel Dekker, Inc., 1985), 116.

¹³¹ Jack P. C. Kleijnen, *Design and Analysis of Simulation Experiments* (New York, NY: Springer Science + Business Media, LLC, 2008), 129.

¹³² A. S. Hedayat, N. J. A. Sloane, and John Stufken, *Orthogonal Arrays: Theory and Applications* (New York, NY: Springer Verlag New York, Inc., 1999), 252.

¹³³ Talbi, 309, 311.

¹³⁴ Torbjörn Lundstedt, Elisabeth Seifert, Lisbeth Abramo, Bernt Thelin, B., Åsa Nyström, Jarle Petterson, and Rolf Bergman, “Experimental Design and Optimization,” *Chemometrics and Intelligent Laboratory Systems* 42, no. 1-2 (1998): 3-40.

heating,¹³⁵ supersonic transport design,¹³⁶ and UAV design.¹³⁷ Despite the breadth of aerospace applications, orthogonal arrays have as of yet to be applied to the optimization of skip entry maneuvers within the current literature. While providing optimal solutions congruent with pseudospectral and meta-heuristic methods, orthogonal arrays permit an augmented exploration of the objective space with the ability to perform main effects analysis.

Methodology

The implementation of the DOE method of orthogonal arrays first requires the formation of the orthogonal array itself. A matrix of dimension $(n \times m)$, an orthogonal array represents a subset of a full-factorial experiment campaign with each row and column corresponding to one experiment and factor (design variable), respectively. Signifying one simulation run, an experiment corresponds to a different combination of factors levels, or design variable values. As an example, the following (2×1) matrix represents two consecutive experiments extracted from an orthogonal array constructed for the present research with six factors (TAV mass, planform area, aerodynamic coefficients, perigee altitude, and bank angle):

$$\begin{bmatrix} m_1 & S_1 & C_{D1} & C_{L1} & h_{p1} & \sigma_1 \\ m_2 & S_2 & C_{D2} & C_{L2} & h_{p2} & \sigma_2 \end{bmatrix} = \begin{bmatrix} 3500 \text{ kg} & 18.9375 \text{ m}^2 & 0.81875 & 0.5000 & 102.2500 \text{ km} & -83.75 \text{ deg} \\ 5500 \text{ kg} & 16.7500 \text{ m}^2 & 1.13750 & 2.0625 & 104.1875 \text{ km} & -82.50 \text{ deg} \end{bmatrix}$$

With the number of experiments as well as the upper and lower bounds for each factor defined as inputs, orthogonal arrays can be constructed using various existing mathematical software suites.

¹³⁵ Kamran Daryabeigi, "Thermal Analysis and Design of Multilayer Insulation for Re-Entry Aerodynamic Heating," *Journal of Spacecraft and Rockets* 39, no. 4 (2002): 509-514.

¹³⁶ Anthony A. Giunta, Vladimir Balabanov, Dan Haim, Bernard Grossman, William H. Mason, Layne T. Watson, and Raphael T. Haftka, "Multidisciplinary Optimization of a Supersonic Transport Using Design of Experiments Theory and Response Surface Modeling," *TR 97-10* (Blacksburg, VA: Virginia Polytechnic Institute and State University, 2001).

¹³⁷ Jeremy S. Agte, Nicholas Borer, and Olivier de Weck, "Design of Long Endurance Systems with Inherent Robustness to Partial Failures during Operations," *Journal of Mechanical Design* 134, no. 10 (2012): 100903-100918.

Once the experiments comprising the orthogonal array are simulated, the resulting objective space can be analyzed in terms of main effects and Pareto optimality. For examples outlining the calculation of main effects for simple orthogonal arrays, see *An Introduction to Design of Experiments: A Simplified Approach* by Barrentine and *Statistical Design of Experiments with Engineering Applications* by Rekab and Shaikh.¹³⁸ For information related to the mathematical theory underpinning orthogonal arrays and main effects analysis, see *Orthogonal Arrays: Theory and Applications* by Hedayat, Sloane, and Stufken.¹³⁹

Fundamentally, Pareto analysis seeks to identify a set of optimal solutions for a given objective space and is utilized for multi-objective optimization within a diverse range of disciplines from economics and management to science and engineering. As stated by Talbi, a solution is considered *Pareto optimal* if it is “not possible to improve a given objective without deteriorating at least [one other] objective” within the MOP.¹⁴⁰ Alternatively, a Pareto optimal solution represents a non-dominated solution within the objective space.¹⁴¹ For each objective space obtained from the experiment campaigns comprising this research, the Pareto optimal set, or *Pareto front*, is determined with a heuristic filter algorithm which identifies solutions as either dominated or non-dominated and discards the former.¹⁴²

¹³⁸ Larry B. Barrentine, *An Introduction to Design of Experiments: A Simplified Approach*, (Milwaukee, WI: Quality Press, 1999); Kamel Rekab and Muzaffar Shaikh, *Statistical Design of Experiments with Engineering Applications* (Boca Raton, FL: Taylor and Francis Group, LLC., 2005).

¹³⁹ Hedayat et al., *Orthogonal Arrays: Theory and Applications*.

¹⁴⁰ Talbi, 308.

¹⁴¹ J. Dario Landa Silva, Edmund K. Burke, and Sanja Petrovic, “An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling,” in *Metaheuristics for Multiobjective Optimization*, ed. Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen, and Vincent T’kindt (Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2004), 96.

¹⁴² J. S. Arora, *Introduction to Optimum Design*, Third Edition (Waltham, MA: Academic Press, 2012), 670.

Since the optimal set of TAV design parameters is unknown, the DOE framework can be employed to create an orthogonal array of experiments formulated according to the notional factor level bounds outlined in Table 5.1 for each design parameter, or factor.

Table 5.1. Factors and Associated Level Bounds for TAV Design Parameters

<i>Factor</i>	<i>Minimum Value</i>	<i>Maximum Value</i>
Mass, kg	2000	6000
Planform Area, m ²	15	22
Drag Coefficient	0.5	2.2
Lift Coefficient	0.5	3.0

With the decision space comprising four TAV design factors and three factors of h_p , h_i , σ related to trajectory design, a systematic optimization approach is required. Initially, two consecutive sets of experiments (Campaigns #1, 2) are conducted to identify appropriate factor bounds on the perigee and initial altitude for the skip entry trajectory, respectively, with $\sigma = -90$ deg. For Campaign #1, $h_p \in [75, 100]$ km with $h_i = 1000$ km; for Campaign #2, the perigee altitude varies according to the preceding campaign results, with $h_i \in [300, 1000]$ km. Remaining at $\sigma = -90$ deg, Campaign #3 is then run to establish an objective space from which Pareto solutions to the three MOPs are identified. Converting the bank angle into an active factor varying within the interval $\sigma \in [-120, 0]$ deg, Campaign #4 produces a set of Pareto solutions which are then compared to those obtained from the preceding campaign to determine the coupled TAV and trajectory design that satisfies the primary MOP.

Results and Analysis

With the experiment campaigns as a foundation, the constant bank angle analysis is first discussed, to include a presentation of the Pareto optimal fronts for the three MOPs as well as a comparison of the main effects and Pareto front analysis for the primary MOP. Following the selection of the optimal TAV and trajectory design based on a comparison of the constant and variable bank angle analysis results, functions for $\Delta V = f(\sigma)$ and $\Delta i = f(\sigma, \Delta V)$ are derived via regression analysis. Finally, the performance of the optimal TAV and trajectory design is compared with that of an exo-atmospheric simple plane change.

Constant Bank Angle Analysis

For Campaign #1, a preliminary orthogonal array with 125 experiments and 5 levels yielded a success rate of 49.6%, with 63 experiments failing since particular combinations of TAV and trajectory factors result in either planetary impact or a failure to establish a stable re-circularized orbit following perigee transit, thus producing an eventual impact scenario. With 62 successful experiments producing a sparse objective space, a higher-density experiment array was desired and, therefore, the number of experiments increased to 3125. After processing the higher-density experiment array, the number of successful experiments increased from 62 to 1575. From the objective space, it was observed that the perigee altitude of 81.25 km represented the lowest of the five levels to produce a successful experiment. Restricted by the number of levels employed to create the experiment array, it was concluded that a proper lower bound for the perigee altitude factor was not 81.25 km, but rather a value between the initial lower bound of 75 km and 81.25 km. Calculating the median of these two values and rounding up to the nearest integer thus produced a new lower bound of 79 km for the perigee altitude factor.

In addition, the perigee altitude upper bound was also modified with the value increasing from 100 km to 110 km. While the upper limit of the sensible atmosphere is defined at an altitude of 120 km, the altitude of 110 km was selected since it corresponds to a calculated atmospheric density of $5.930 \times 10^{-8} \text{ kg/m}^3$, a value 300% greater than the density of $1.474 \times 10^{-8} \text{ kg/m}^3$ at 120 km. With a greater atmospheric density, an altitude of 110 km permits an increased ability of a given TAV design to perform an out-of-plane maneuver during a banked skip entry and thus achieve a change in maximum orbit inclination, albeit small in magnitude. Also constructed with 3125 experiments and 5 levels, Campaign #2 was run with $h_p \in [79, 110] \text{ km}$ and produced a success rate of 54.8% with trajectory solution distribution conforming to the *a priori* expectation that as the initial altitude increases, the likewise increase in orbital potential energy contributes to an increase in the maximum inclination change. As a result, the initial altitude was set to 1000 km for the remaining experiment campaigns in order to satisfy the primary MOP for maximizing Δi .

When executed, Campaign #3 (3125 experiments, 5 levels) produced the first objective space from which a Pareto optimal front could be determined based on the primary MOP in Eq. (5.1). From the 2138 successful experiments, 10 were identified as being non-dominated and comprising the Pareto optimal set as shown in Fig. 5.1.

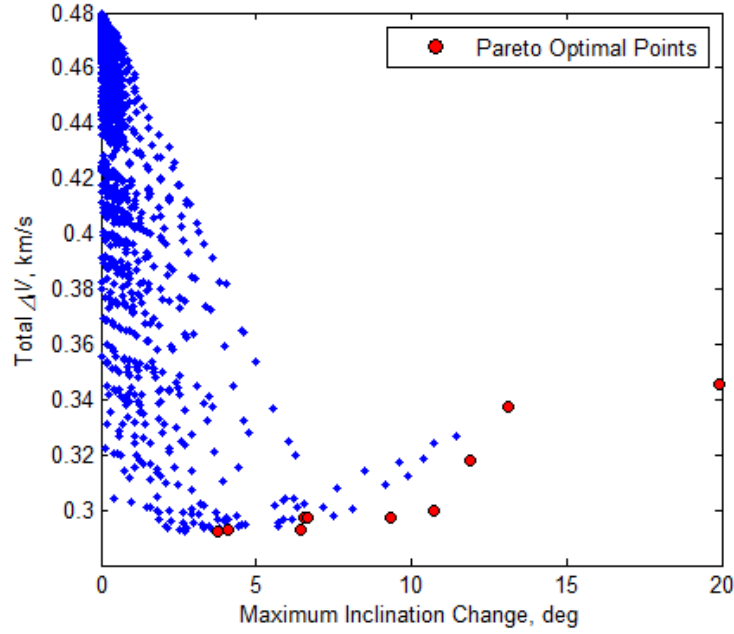


Figure 5.1. Pareto Optimal Front for Campaign #3: $\{\max(\Delta i), \min(\Delta V_{Total})\}$

The Pareto optimal fronts related to the MOPs in Eqs. (5.2) and (5.3) are given in Figs. 5.2 and 5.3, respectively:

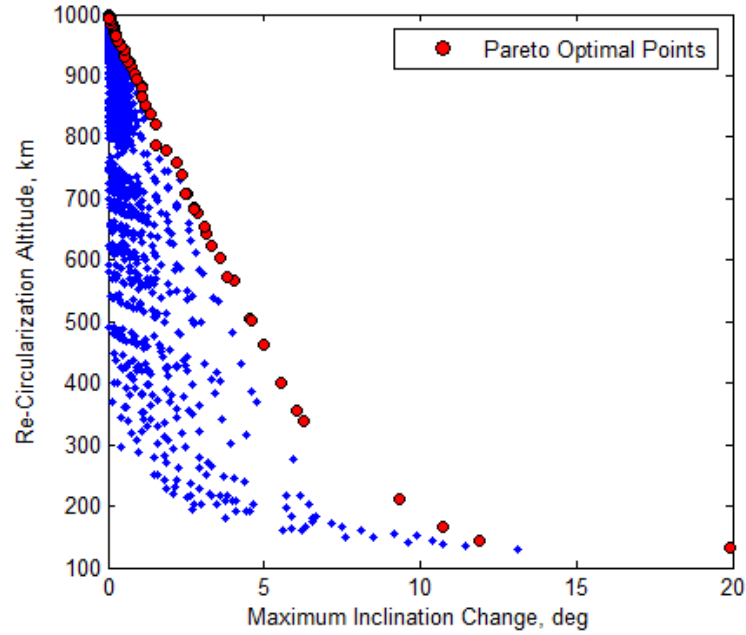


Figure 5.2. Pareto Optimal Front for Campaign #3: $\{\max(\Delta i), \max(h_{recirc})\}$

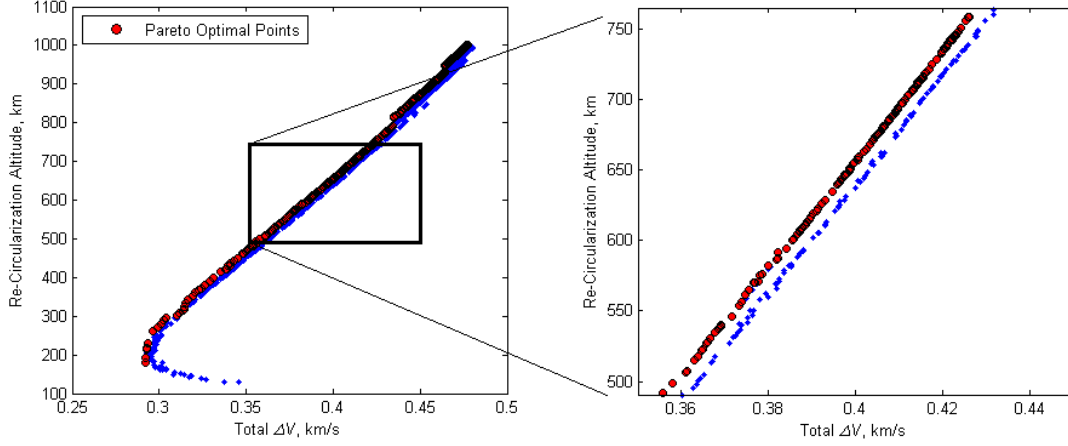


Figure 5.3. Pareto Optimal Front for Campaign #3: $\{\min(\Delta V), \max(h_{recirc})\}$

One data analysis option available for determining the optimal TAV and trajectory designs is to map the Pareto optimal set from the objective space ΔV vs. Δi in Fig. 5.1 onto the objective spaces h_{recirc} vs. Δi and h_{recirc} vs. ΔV in Figs. 5.2 and 5.3, respectively. Shown in Fig. 5.4, the Pareto optimal set identified with circles in Fig. 1 is mapped to the squares in Fig. 5.4(a). Upon comparison, the two sets of Pareto optimal points yield a set of intersecting points which satisfy both the primary MOP and the secondary MOP of $\{\max(\Delta i), \min(h_{recirc})\}$. When mapped to the objective space in Fig. 5.4(b), however, the set of intersecting Pareto points do not coincide with any of the Pareto points satisfying the tertiary MOP of $\{\min(\Delta V), \max(h_{recirc})\}$. While non-intersection persists in Subplot (b) when the boundaries $\{\max(\Delta V), \max(h_{recirc})\}$ and $\{\min(\Delta V), \min(h_{recirc})\}$ are plotted, a single point of intersection does arise for the boundary representing $\{\max(\Delta V), \min(h_{recirc})\}$ – a non-optimal flight condition. Overall, Pareto intersection analysis produces four candidate TAV designs which maximize Δi , while minimizing both the total ΔV and re-circularization altitude. Since these designs do not maximize re-circularization altitude while minimizing total ΔV , subsequent analysis is restricted to satisfying only the primary MOP.

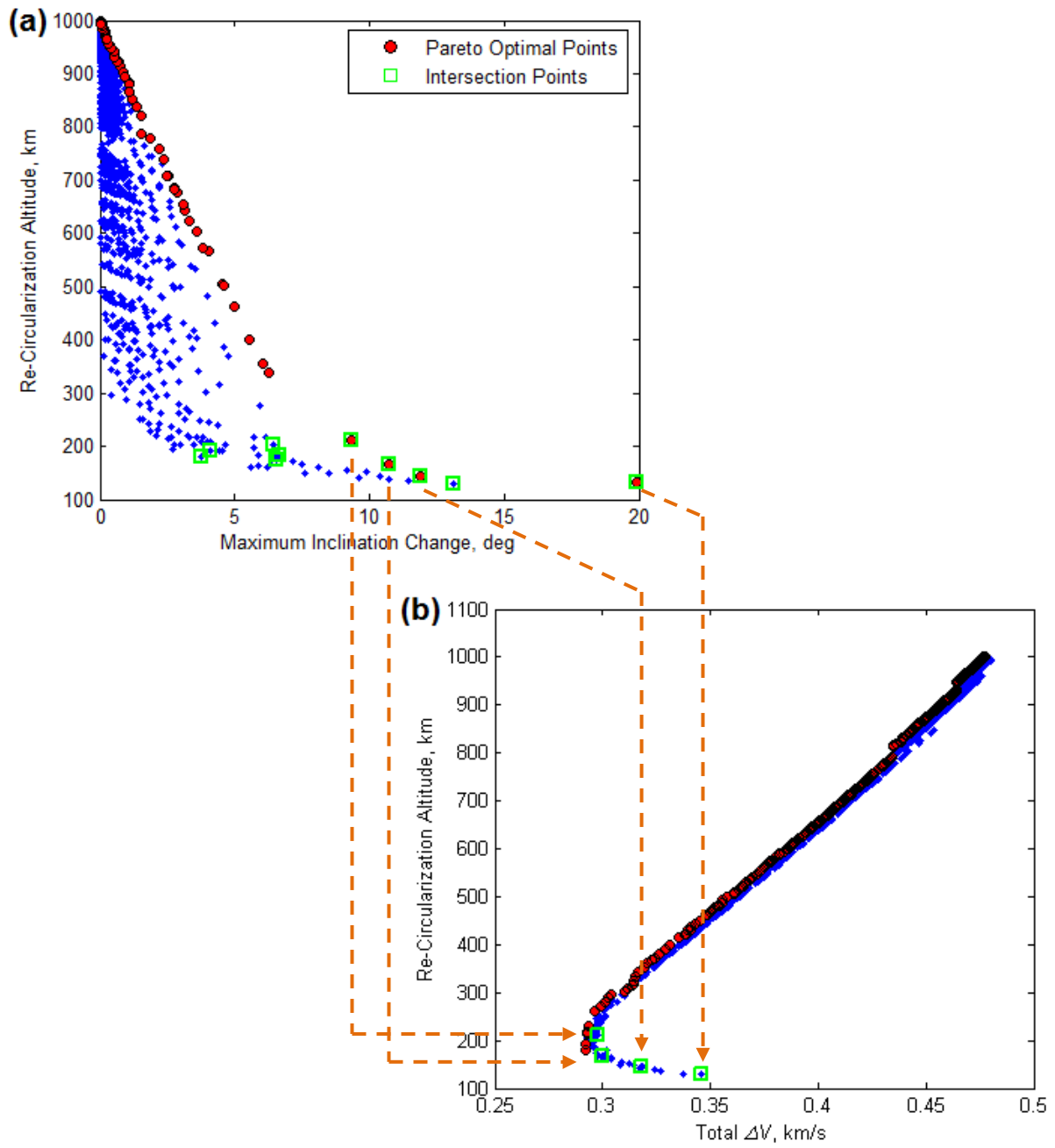


Figure 5.4. Mapping of Pareto Optimal Set from ΔV vs. Δi onto Secondary and Tertiary Objective Spaces

Besides forming objective spaces, the solutions obtained from the orthogonal array experiment campaigns can also be employed to calculate the main effects of each factor on a selected measure of performance for the system. Although the MOP is defined as the simultaneous optimization of Δi and ΔV , the former can be viewed as a primary driver of TAV and trajectory optimization based on the exigencies of immediate mission requirements. During nominal mission operations, ΔV performance becomes essential since vehicle mission longevity is irrevocably contingent on propellant availability. Focusing on the maximization of Δi , Fig. 5.5 depicts the main effects of the TAV design factors on Δi .

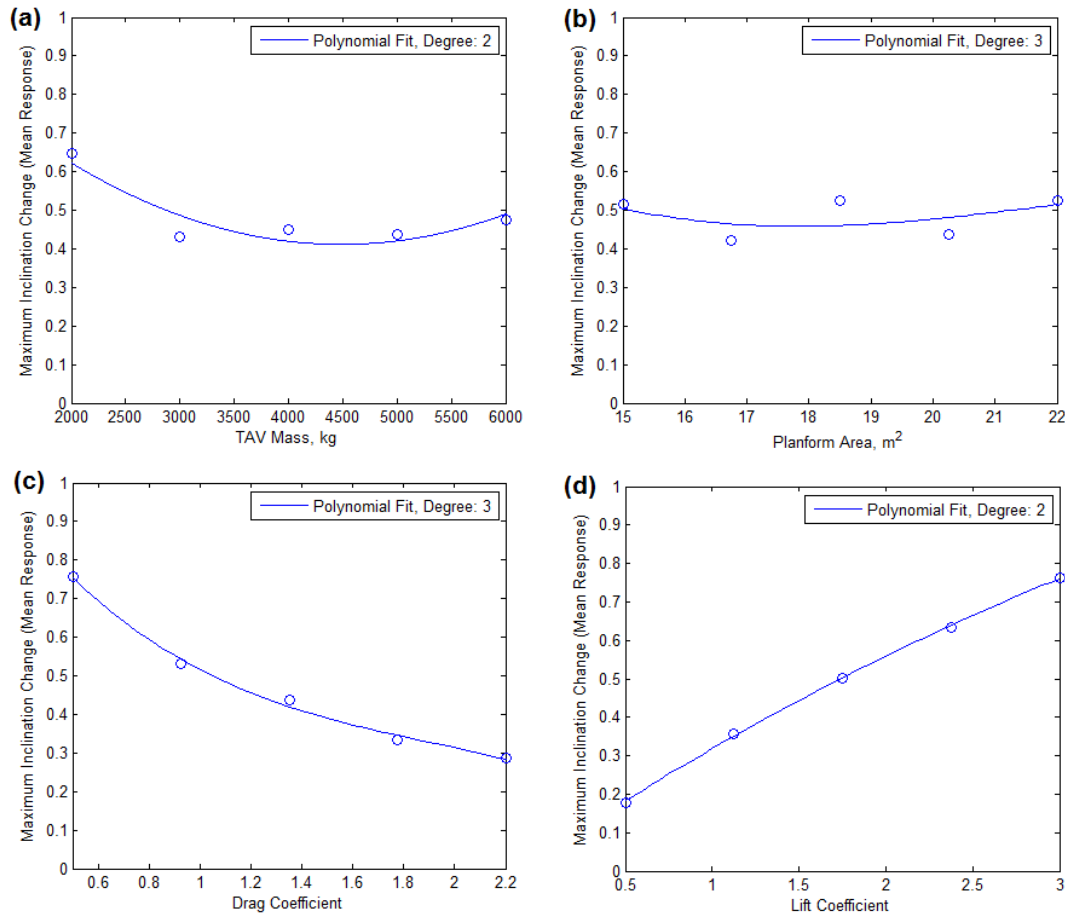


Figure 5.5. Main Effect on Maximum Inclination Change for DOE Campaign #3 with
(a) TAV Mass, (b) Planform Area, (c) Drag Coefficient, and (d) Lift Coefficient

Referencing Fig. 5.5(a) as an example, the main effect of TAV mass on maximum inclination change, at a sample level of 4000 kg, is the mean inclination change of all factors within the orthogonal array with the TAV mass equal to 4000 kg. Graphically, the number of discrete points in each subplot in Fig. 5.5 is equal to the number of levels for each factor within the orthogonal array. When plotted, the slope of the points as well as any curve fits indicates the relative strength of the main effect on the desired measure of performance. Of the TAV design factors, the drag and lift coefficients produce the greatest relative slopes and, therefore, are considered to contribute the greatest influence on maximum inclination change; nearly horizontal in slope, planform area has the least influence. In order to maximize Δi , the main effects from Fig. 5.5 coalesce to form a potential TAV design with $m = 2000$ kg, $C_D = 0.5$, and $C_L = 3.0$. Based on the approximate horizontal distribution of the planform area main effects, the mean of the planform area decision space of $S = 18.5 \text{ m}^2$ is selected as the final component of the potential TAV design.

Similar to the TAV design factors, the main effects of the perigee altitude factor can also be calculated and plotted (see Fig. 5.6). Of the five design factors evaluated within the orthogonal array, perigee altitude features the greatest comparative impact on maximum inclination change. At the minimum factor bound of $h_p = 79$ km the mean response is 6 deg – a value one order of magnitude greater than the mean response of 0.75 deg obtained from the main effect plots for the aerodynamic coefficients. As expected, Fig. 5.6 illustrates that as perigee altitude decreases, the ability of a TAV to perform aeroassisted out-of-plane maneuvers increase due to the exponential increase in atmospheric density.

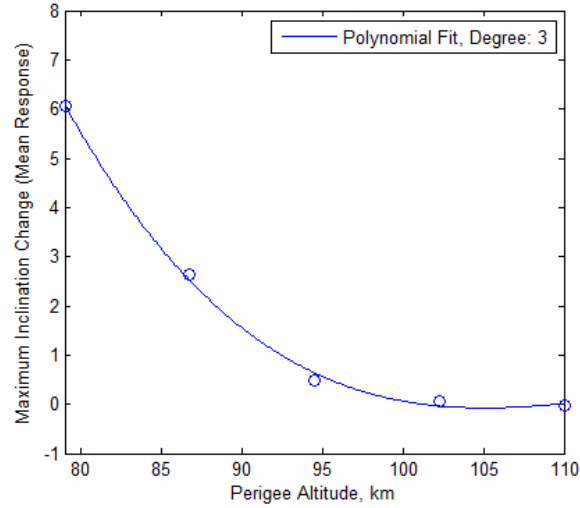


Figure 5.6. Main Effect of Perigee Altitude on Maximum Inclination Change for DOE Campaign #3

With a potential TAV and trajectory design established from the main effects analysis, the optimality of the design in terms of the MOP must be evaluated through a comparison with the Pareto optimal set obtained from the ΔV vs. Δi objective space. Prior to any comparison, two supplementary experiment campaigns were run in an effort to populate the sparse objective space in the range $10 \text{ deg} \leq \Delta i \leq 20 \text{ deg}$. Shown in Table 5.2, the first of the supplemental campaigns focused on exploring the decision space arising from the Pareto optimal set, while the second was more limited and focused on two outlier points observed from preliminary inspections of the objective space. Of these outlier points, the first corresponded to the Pareto optimal solution which yielded $\Delta i = 19.91 \text{ deg}$ for $\Delta V = 0.345 \text{ km/s}$ as shown in Fig. 5.1. For the second point, outlier status was assigned not for inclination change performance, but rather the maximum deceleration and stagnation heat flux experienced during skip entry. While the solutions comprising the objective space maintained an average deceleration and heat flux of 0.17 g and 129.20 kW/m^2 , the identified outlier point was considerably higher with 5.56 g and 1351.5 kW/m^2 .

Table 5.2. Factors and Associated Level Bounds for Supplementary DOE Campaigns

<i>Factor</i>	<i>Campaign</i>	
	<i>1</i>	<i>2</i>
Mass, kg	[2000, 5000]	[2000, 2000]
Planform Area, m ²	[15, 22]	[18, 22]
Drag Coefficient	[0.5, 0.5]	[0.5, 0.5]
Lift Coefficient	[2.0, 3.0]	[0.5, 3.0]
Perigee Altitude, km	[86, 87]	[86, 87]

Due to the restricted decision space of the design factors, low-density orthogonal arrays of 125 experiments were constructed for each of the supplementary campaigns. With success rates of 36% each, these campaigns further populated the ΔV vs. Δi objective space from Fig. 5.1 and, as a result, created an augmented Pareto optimal front based on the addition of more solutions to the objective space as shown in Fig. 5.7.

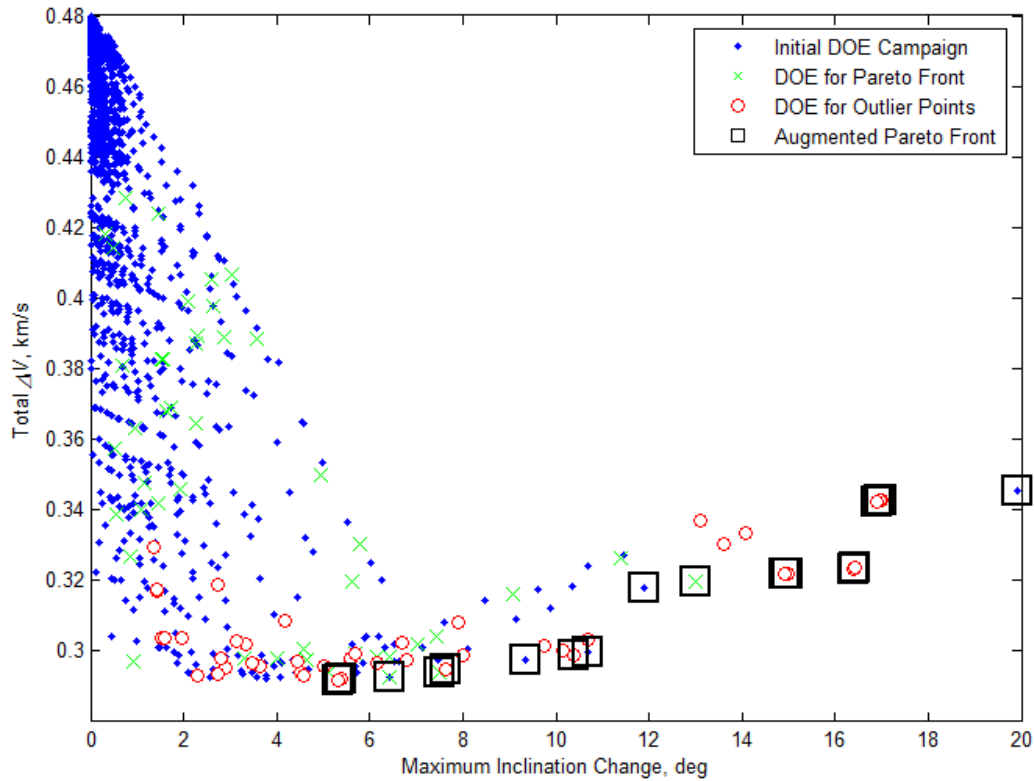


Figure 5.7. Augmented Pareto Optimal Front for DOE Campaign #3

Demarcated by squares in Fig. 5.7, the augmented Pareto optimal set is given in Table 5.3 in ascending order of maximum inclination change with associated TAV and trajectory design factors, as well as values related to maximum deceleration and stagnation heat flux performance. From the Pareto optimal set, the TAV design which produced the greatest change in inclination ($\Delta i = 19.91$ deg) is identical to that estimated through the main effects analysis, with $m = 2000$ kg, $S = 18.5$ m², $C_D = 0.5$, and $C_L = 3.0$. The only design difference arises from the trajectory, with the main effects and Pareto front analyses yielding perigee altitudes of 86.75 km and 79 km, respectively.

Table 5.3. Maneuver Parameters of Augmented Pareto Optimal Front

<i>Mass, kg</i>	<i>Planform Area, m²</i>	<i>C_D</i>	<i>C_L</i>	<i>Perigee, km</i>	<i>Δi, deg</i>	<i>Max. Decel., g</i>	<i>Q̇_{s,max}, kW/m²</i>
2000	19.00	0.500	1.125	86.50	5.31	0.17	370.48
2000	18.00	0.500	1.125	86.25	5.38	0.17	380.81
2000	18.50	0.500	1.750	86.75	6.42	0.21	363.91
2750	22.00	0.500	1.750	86.00	7.49	0.22	396.45
2000	19.00	0.500	1.750	86.75	7.64	0.22	364.00
2000	16.75	0.500	3.000	86.75	9.34	0.35	367.01
2000	18.00	0.500	3.000	87.00	10.38	0.35	356.20
2000	18.50	0.500	2.375	86.75	10.70	0.30	364.48
4000	22.00	0.925	2.375	86.75	11.88	0.17	360.97
3500	16.75	0.875	3.000	86.00	12.99	0.22	395.01
2000	18.00	0.500	2.375	86.50	14.90	0.30	373.39
2000	19.00	0.500	2.375	86.75	14.98	0.30	363.20
2000	18.00	0.500	3.000	86.75	16.38	0.37	363.96
2000	19.00	0.500	3.000	87.00	16.43	0.37	354.04
2000	20.00	0.500	1.750	86.75	16.89	0.23	362.52
2000	18.00	0.500	1.750	86.25	16.95	0.23	382.86
2000	19.00	0.500	1.750	86.50	16.98	0.23	372.45
2000	18.50	0.500	3.000	86.75	19.91	0.38	362.96

Variable Bank Angle Analysis

From the main effects and Pareto front analysis of Campaign #3, a candidate design for TAV and skip trajectory was generated for a bank angle of $\sigma = -90$ deg. So as to ensure the optimality of the candidate design, Campaign #4 was conducted to ascertain if $\sigma = -90$ deg satisfies the MOP by introducing bank angle as a sixth factor which varies within $\sigma \in [-120, 0]$ deg. Overall, five orthogonal arrays (729 experiments, 9 levels each) were created with the TAV design parameters from Table 5.1 and $h_p \in [79, 110]$ km as the baseline factors, and bank angle varying according to the following: $\sigma \in [-120, -100]$ deg, $\sigma \in [-100, -80]$ deg, $\sigma \in [-80, -50]$ deg, $\sigma \in [-50, -20]$ deg, and $\sigma \in [-20, 0]$ deg. When combined, the solutions from each of the five orthogonal arrays produced the objective space illustrated in Fig. 5.8 with a success rate of 75.8% from the 3645 total experiments.

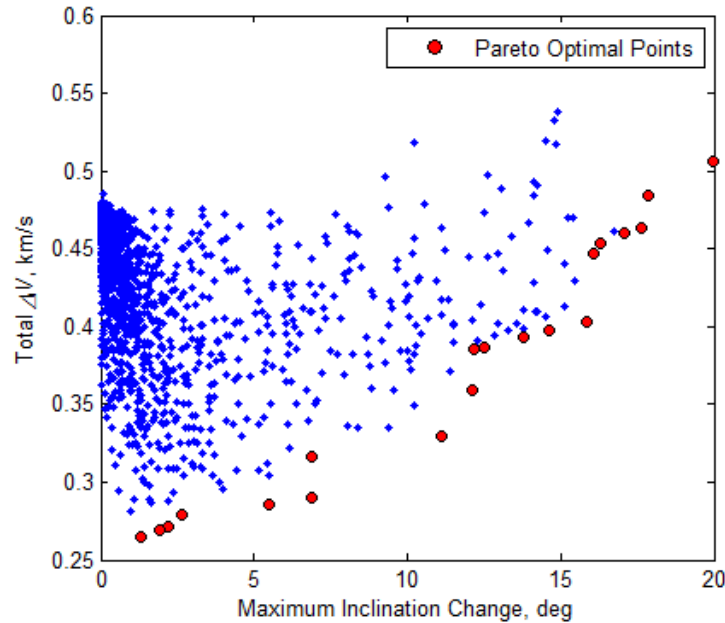


Figure 5.8. Pareto Optimal Front for DOE Campaign #4: $\{\max(\Delta i), \min(\Delta V_{Total})\}$

A total of 20 points, the Pareto optimal set from Campaign #4 in Fig. 5.8 is listed in Table 5.4 in ascending order of maximum inclination change. When compared with the objective space from the preceding campaign, the Pareto optimal set from Campaign #4 yielded a TAV and trajectory design which achieved a 0.05 deg greater inclination change of $\Delta i = 19.96$ deg at $\sigma = -57.50$ deg and $h_p = 79$ km. By examining required ΔV , however, the hundredths-place increase to inclination change corresponds to $\Delta V = 0.51$ km/s, an increase of 47.8% in ΔV expenditure from Campaign #3 for $\Delta i = 19.91$ deg. In addition to requiring a higher ΔV , the design also features a maximum deceleration greater than 1.0 g and a maximum stagnation heat flux nearly double the value calculated for the design from Campaign #3.

Table 5.4. Maneuver Parameters of Pareto Optimal Front for DOE Campaign #4

<i>Mass, kg</i>	<i>Planform Area, m²</i>	<i>C_D</i>	<i>C_L</i>	<i>Perigee, km</i>	<i>σ, deg</i>	<i>Δi, deg</i>	<i>Max. Decel., g</i>	<i>Q̇_{s,max}, kW/m²</i>
4500	15.88	0.500	0.500	82.88	-115.00	1.30	0.17	528.35
2000	17.63	1.988	3.000	94.50	-107.50	1.91	0.17	162.59
3500	17.63	1.775	2.375	90.63	-102.50	2.17	0.17	247.41
4000	20.25	1.775	3.000	90.63	-107.50	2.66	0.17	225.41
6000	17.63	1.563	1.125	86.75	-112.50	5.47	0.17	435.18
2000	17.63	1.350	1.125	90.63	-120.00	6.87	34.34	1471.03
5000	15.00	0.925	0.813	82.88	-76.25	6.89	0.17	537.23
5000	17.63	0.713	3.000	82.88	-85.00	11.09	0.32	543.80
4500	21.13	0.713	1.125	82.88	-68.75	12.12	0.17	546.48
6000	20.25	1.775	1.750	82.88	-20.00	12.19	0.24	529.81
2000	19.38	2.200	2.063	90.63	-15.00	12.49	0.20	259.43
3500	16.75	2.200	2.063	86.75	-20.00	13.80	0.20	371.35
4000	19.38	2.200	2.688	86.75	-38.75	14.63	0.23	369.96
3500	22.00	1.775	2.688	86.75	-50.00	15.84	0.27	366.78
5000	19.38	0.925	2.063	79.00	-38.75	16.06	0.46	710.99
6000	20.25	1.138	2.688	79.00	-38.75	16.32	0.53	712.46
5500	22.00	0.925	2.688	79.00	-46.25	17.07	0.62	715.91
4500	17.63	0.925	3.000	79.00	-50.00	17.63	0.67	714.22
3000	22.00	0.500	2.063	79.00	-50.00	17.84	0.88	721.67
2500	20.25	0.500	3.000	79.00	-57.50	19.96	1.41	719.15

When plotted, the main effects for each factor reveal greater dynamism than shown for Campaign #3. So as to maximize Δi , the main effects of the subplots in Fig. 5.9 create a potential TAV design with $m = 2000$ kg, $C_D = 0.5$, and $C_L = 3.0$. With the main effects point distribution for planform area producing a cubic curve fit, the approximate local maxima of the planform area decision space of $S = 20.25$ m² is selected to complete the TAV design. Although Fig. 5.10(a) mirrors the same trend for perigee altitude from Campaign #3, Subplot (b) indicates a local maximum change in inclination for a bank angle of approximately $\sigma = -20$ deg.

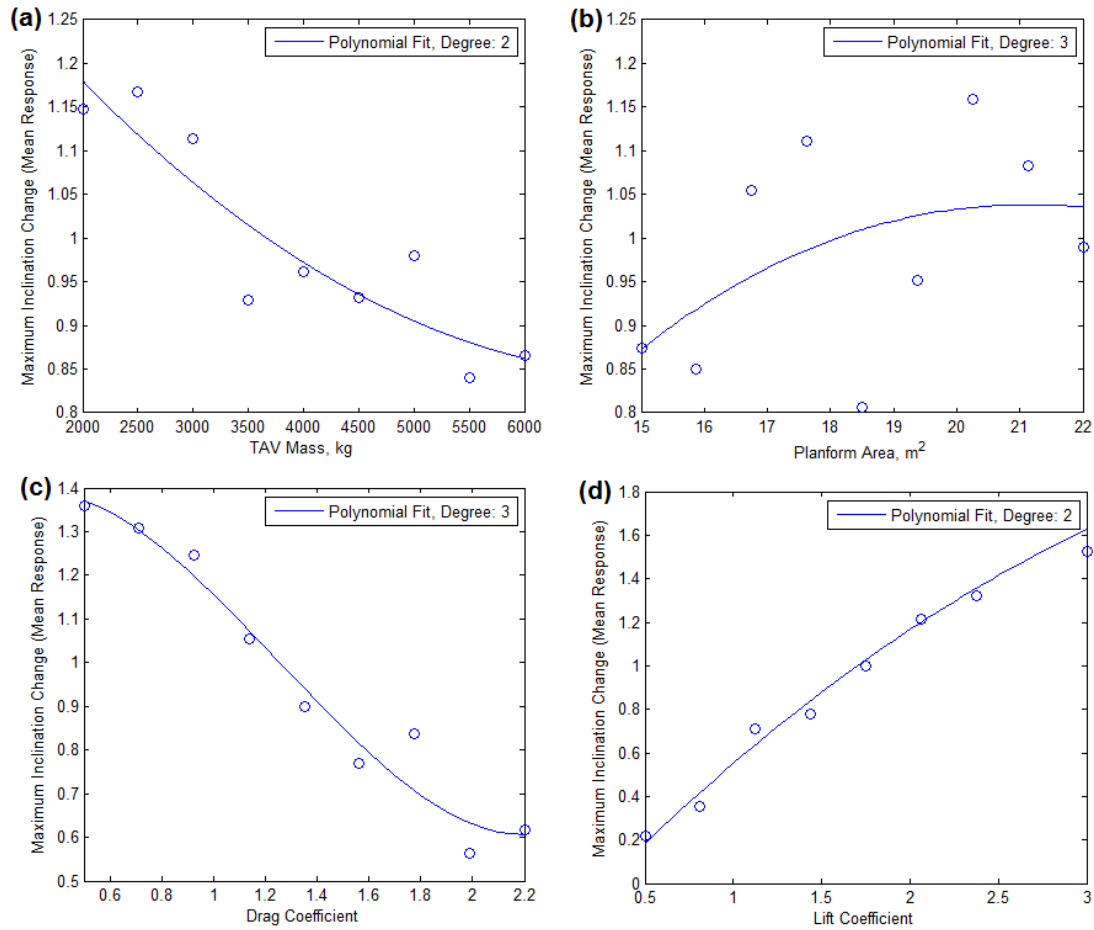


Figure 5.9. Main Effect on Maximum Inclination Change for DOE Campaign #4 with (a) TAV Mass, (b) Planform Area, (c) Drag Coefficient, and (d) Lift Coefficient

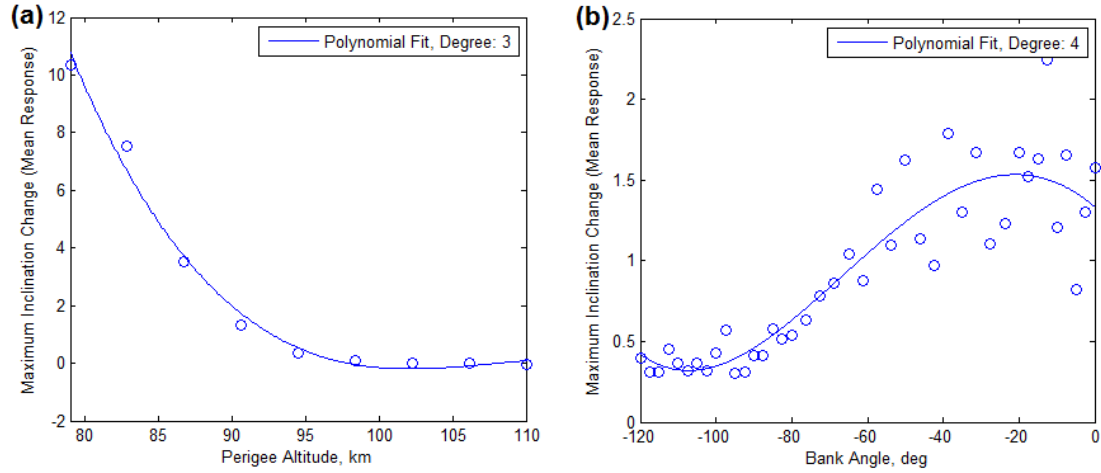


Figure 5.10. Main Effect on Maximum Inclination Change for DOE Campaign #4 with (a) Perigee Altitude, and (b) Bank Angle

Unlike Campaign #3, disparities arise in the TAV and trajectory design when the results of the main effects and Pareto front analysis are compared from Campaign #4. Even though aligning for the factors of planform area, perigee altitude, and the aerodynamic coefficients, the two methods differ for TAV mass and bank angle. Based on the higher ΔV , coupled with the greater maximum deceleration and stagnation heat flux of the design from Campaign #4, the potential TAV and trajectory design from Campaign #3 is thus deemed optimal by satisfying both aspects of the MOP $\{\max(\Delta i), \min(\Delta V_{Total})\}$ and is shown in Table 5.5.

Table 5.5. Optimal TAV Design and Trajectory

Mass, kg	2000
Planform Area, m ²	18.5
Drag Coefficient	0.5
Lift Coefficient	3.0
Initial Altitude, km	1000
Perigee Altitude, km	86.75
Bank Angle	−90 deg

Single TAV Design Analysis

With the optimization phase completed, a fifth experiment campaign was formulated to determine how ΔV and Δi changes as bank angle varies within the interval $\sigma \in [-120, 0]$ deg for single TAV and trajectory design. Composed of 241 experiments incremented at $\sigma = 0.5$ deg, the single TAV campaign resulted in a success rate of 62.6% with the ΔV vs. Δi objective space and accompanying Pareto optimal front shown in Fig. 5.11:

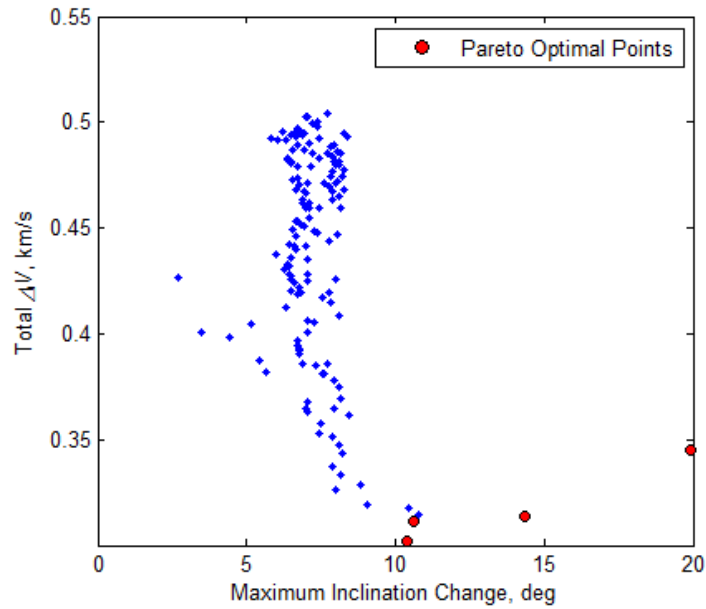


Figure 5.11. Pareto Optimal Front for Single TAV Design: $\{\max(\Delta i), \min(\Delta V_{Total})\}$

Although depicting the graphical relationship between ΔV and Δi , Fig. 5.11 fails to convey the impact of the independent variable σ on these trajectory performance measures. Therefore, the objective space was re-plotted with bank angle as the independent variable in Fig. 5.12.

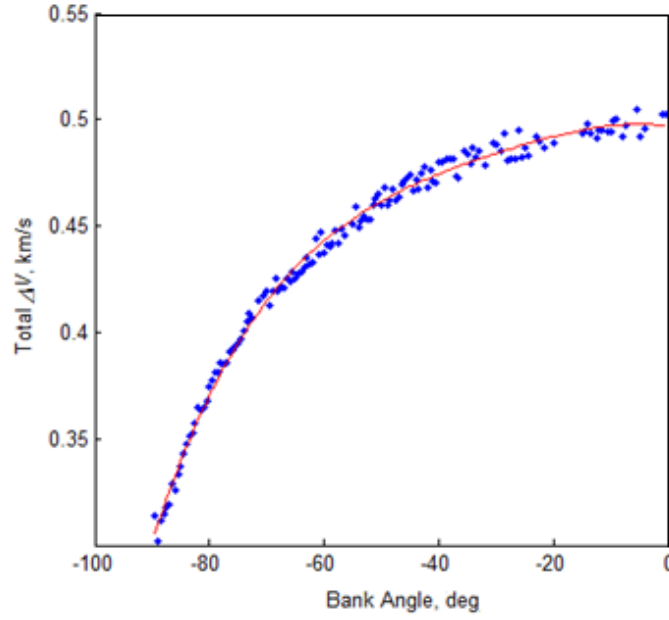


Figure 5.12. Polynomial Fit for Single TAV Design with $\Delta V = f(\sigma)$

Based on the grouping of solutions within the ΔV vs. σ objective space, a univariate quartic polynomial curve fit with constant coefficients was devised. The general expression for a polynomial is given by Eq. (5.4a), while the objective space-specific expression is given by Eq. (5.4b):¹⁴³

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0 \quad (5.4a)$$

$$\Delta V = f(\sigma) = a_4 \sigma^4 + a_3 \sigma^3 + a_2 \sigma^2 + a_1 \sigma + c_0 \quad (5.4b)$$

where

<i>Coefficient</i>	<i>Coefficient Value</i>	<i>95% Confidence Bounds</i>
a_0	0.497	[0.4924, 0.5015]
a_1	-3.867×10^{-4}	$[-1.036 \times 10^{-3}, 2.627 \times 10^{-4}]$
a_2	-4.577×10^{-5}	$[-7.333 \times 10^{-5}, -1.821 \times 10^{-5}]$
a_3	-8.323×10^{-7}	$[-1.272 \times 10^{-6}, -3.922 \times 10^{-7}]$
a_4	-7.106×10^{-9}	$[-9.458 \times 10^{-9}, -4.753 \times 10^{-9}]$

¹⁴³ Edward J. Barbeau, *Polynomials* (New York, NY: Springer-Verlag New York, Inc., 1989), 1.

The square of the correlation coefficient r is defined as:¹⁴⁴

$$r^2 = \frac{ss_{xy}^2}{ss_{xx}ss_{yy}} = \frac{(\sum xy - n\bar{x}\bar{y})^2}{(\sum x^2 - n\bar{x}^2)(\sum y^2 - n\bar{y}^2)} \quad (5.5)$$

where ss_{xx} , ss_{yy} , and ss_{xy} are sum of squared values for a set of n points. For the polynomial model of the ΔV vs. σ objective space, the squared correlation coefficient is computed to be $r^2 = 0.994$ with residuals shown in Fig. 5.13.

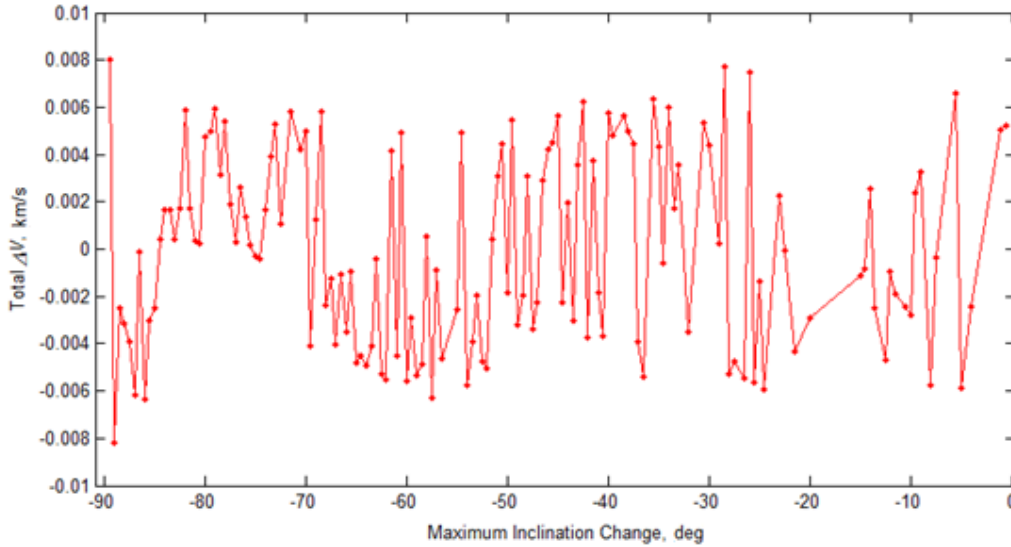


Figure 5.13. Residuals Plot of Polynomial Fit for Single TAV Design with $\Delta V = f(\sigma)$

The objective space from Fig. 5.12 can be expanded with introduction of Δi as the third orthogonal axis. When plotted within three-dimensional space, the new objective space permits the creation of a surface fit with σ and ΔV as function inputs: $\Delta i = f(\sigma, \Delta V)$. Modeled as a bivariate cubic polynomial with constant coefficients, the surface fit is shown graphically in Fig. 5.14 and given symbolically by Eqs. (5.6a) and (5.6b).¹⁴⁵

¹⁴⁴ Eric W. Weisstein, *CRC Concise Encyclopedia of Mathematics*, Second Edition (Boca Raton, FL: CRC Press LLC, 2003), 568.

¹⁴⁵ Keith O. Geddes, Stephen R. Czapor, George Labahn, *Algorithms for Computer Algebra* (Norwell, MA: Kluwer Academic Publishers, 1992), 46.

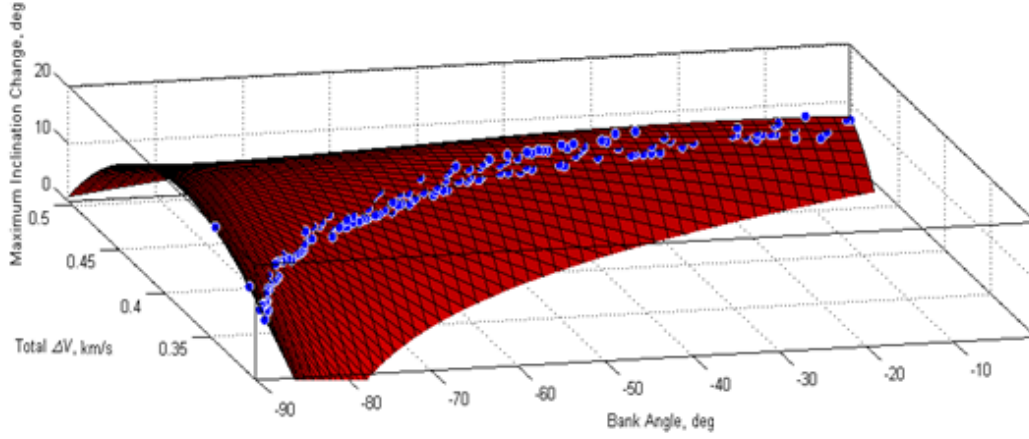


Figure 5.14. Surface Fit for Single TAV Design with $\Delta i = f(\sigma, \Delta V)$

$$f(x, y) = a_{n,m}x^n y^m + \dots + a_{22}x^2 y^2 + a_{21}x^2 y + a_{12}x y^2 + a_{11}x y + a_{10}x + a_{01}y + \quad (5.6a)$$

$$\Delta i = f(\sigma, \Delta V) = a_{03}(\Delta V)^3 + a_{21}(\sigma)^2(\Delta V) + a_{12}(\sigma)(\Delta V)^2 + a_{11}(\sigma)(\Delta V) + a_{20}(\sigma)^2 + a_{02}(\Delta V)^2 + a_{10}(\sigma) + a_{01}(\Delta V) + a_{00} \quad (5.6b)$$

where

<i>Coefficient</i>	<i>Coefficient Value</i>	<i>95% Confidence Bounds</i>
a_{00}	-1983	$[-2347, -1620]$
a_{10}	-22.86	$[-26.86, -18.86]$
a_{01}	7534	$[5692, 9376]$
a_{20}	-0.02106	$[-0.03121, -0.01091]$
a_{02}	-6618	$[-9771, -3465]$
a_{11}	81.95	$[67.68, 96.21]$
a_{21}	0.03915	$[0.01931, 0.5899]$
a_{12}	-72.8	$[-85.47, -60.13]$
a_{03}	-976.5	$[-2842, 888.6]$

Regarding goodness of fit, the model for $\Delta i = f(\sigma, \Delta V)$ features $r^2 = 0.979$, a sum-squared error of $SSE = 6.37$ computed from Eq. (5.7), and a three-dimensional plot of residuals shown in Fig. 5.15.¹⁴⁶

$$SSE = ss_{yy}(1 - r^2) \quad (5.7)$$

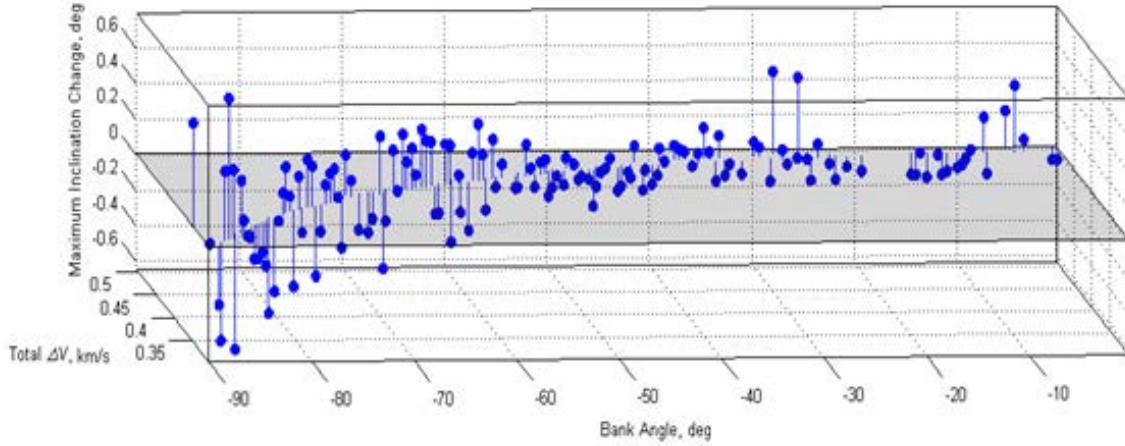


Figure 5.15. Residuals Plot of Surface Fit for Single TAV Design with $\Delta i = f(\sigma, \Delta V)$

Due to the narrow distribution of solutions within the three-dimensional objective space of Fig. 5.14, the surface fit model described by Eq. (5.6b) contains a limited domain. As an example, function inputs of $\sigma = 85$ deg and $\Delta V = 0.337$ km/s will produce a value for Δi corresponding to a three-dimensional section of points comprising the objective space. With function inputs of $\sigma = 85$ deg and $\Delta V = 0.427$ km/s, then the resulting Δi is incorrect since it resides outside of the objective space. Consequently, Eqs. (5.4b) and (5.6b) must be employed sequentially, with bank angle and the function output of Eq. (5.4b) serving as the inputs to the function given by Eq. (5.6b). When the surface fit model is solved accordingly, a three-dimensional solutions curve of the objective space is produced as illustrated by Fig. 5.16.

¹⁴⁶ Weisstein, 568.

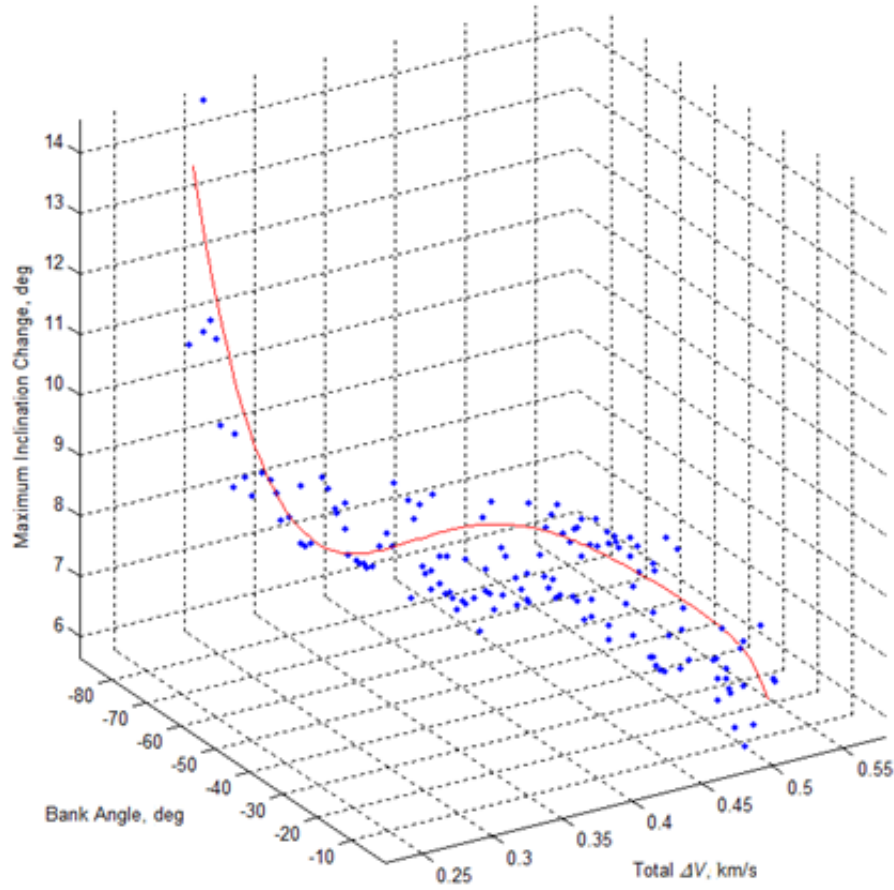


Figure 5.16. Three-Dimensional Solution for Single TAV Design with $\sigma \in [-120, 0]$ deg, $\Delta V = f(\sigma)$, and $\Delta i = f(\sigma, \Delta V)$

Aligning with the results from Campaign #3, an analysis of the h_{recirc} vs. Δi objective space for the single TAV design indicates that the re-circularization altitude decreases as the inclination change increases. This trend is valid since the deeper penetration of the TAV into the atmosphere increases the amount of inclination change, which, in turn, decreases both the orbital energy and re-circularization altitude. Plotted in Fig. 5.17, the re-circularization altitudes vary from 131.2 km to 789.4 km for $\sigma \in [-120, 0]$ deg and $h_i = 1000$ km. Although lucrative for certain mission taskings, the maximum inclination change of $\Delta i = 19.91$ deg is detrimental to the prospect of continued orbital operations since re-circularization occurs at the skip apogee

altitude of 131.2 km. Alternatively, one option available to achieve a high inclination change and regain orbital energy is to perform a maneuver which re-circularizes the trajectory at an orbital altitude higher than skip apogee. Shown in Fig. 5.18(b), the ΔV vs. Δi objective space reflects the completion of a Hohmann transfer up to an example altitude of 500 km for all skip entry trajectories resulting in a skip apogee less than 500 km. While $\Delta i = 19.91$ deg is still achievable, the combined orbit raising and re-circularization increases the total ΔV expenditure for the skip entry by 133.6%, from $\Delta V = 0.345$ km/s in Fig. 5.18 (a), to $\Delta V = 0.806$ km/s in Subplot (b).

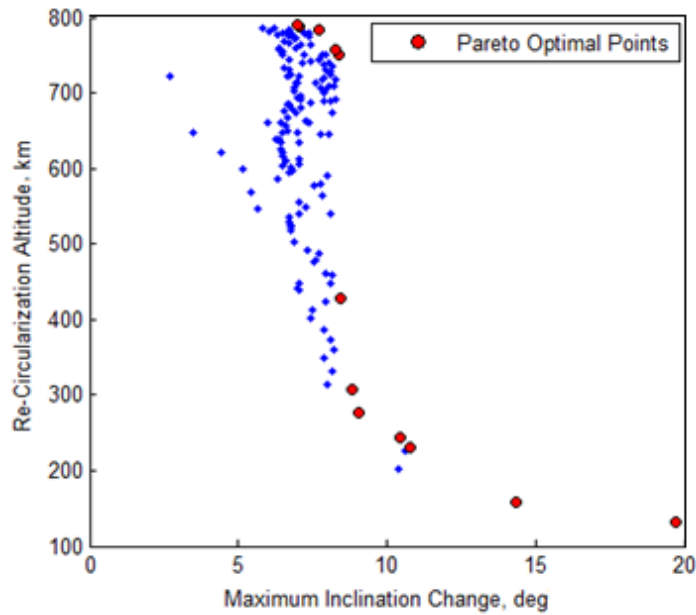


Figure 5.17. Pareto Optimal Front for Single TAV Design: $\{\max(\Delta i), \max(h_{recirc})\}$

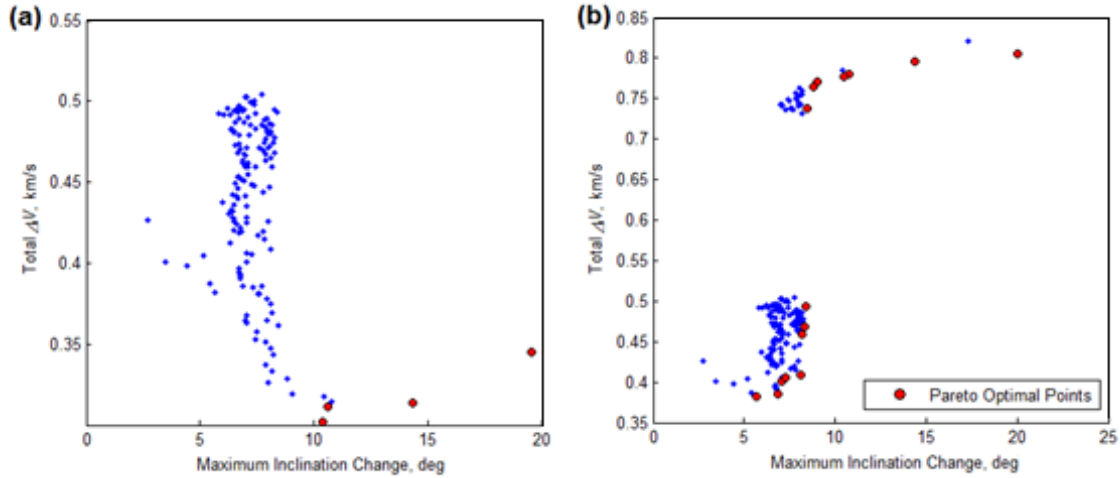


Figure 5.18. Pareto Optimal Fronts for Single TAV Design with (a) Re-Circularization at Skip Apogee, and (b) Re-Circularization at $h = 500$ km via Hohmann Transfer

TAV Design Application

Employing the optimal TAV design listed in Table 5.5, the optimal skip entry trajectory was simulated with respect to a circular reference orbit with the following initial states:

Table 5.6. Reference Orbit Initial States for Optimal Design Simulation

Eccentricity	0.0
Altitude, km	1000
Longitude	0 deg
Latitude	0 deg
Inclination Angle	37.84 deg

Depicted by a solid line in Fig. 5.19, the reference orbit maintains a maximum orbit inclination of 37.84 deg, the result of a due East launch from Wallops Island, VA. So as to achieve $\Delta i = 19.91$ deg, the TAV must reach a perigee altitude of 86.75 km during skip entry. Also referred to as a perturbed orbit, the initial states for the skip entry trajectory are given in Table 5.7. For both the reference and perturbed orbits, trajectory time is measured as an elapsed quantity from $t = 0$ at the initial longitude/latitude coordinates $(\theta_i, \phi_i) = (0, 0)$ deg.

Table 5.7. Perturbed Orbit Initial States for Optimal Design Simulation

Altitude, km	1000
Longitude	0 deg
Latitude	0 deg
Inclination Angle	37.84 deg
Flight-Path Angle	0 deg
Heading Angle	37.84 deg
Bank Angle	-90 deg

Even though $\sigma = -90$ deg at the start of the simulation, a shifting in the perturbed orbit with respect to the reference orbit does not occur until the TAV approaches the upper limit of the sensible atmosphere at an altitude of 120 km and descends below it. As shown in Fig. 5.19, the perturbed orbit begins to shift at an approximate longitude of 140 deg E and reaches the first instance of maximum inclination deviation with the reference orbit at $\theta \approx 45$ deg E.

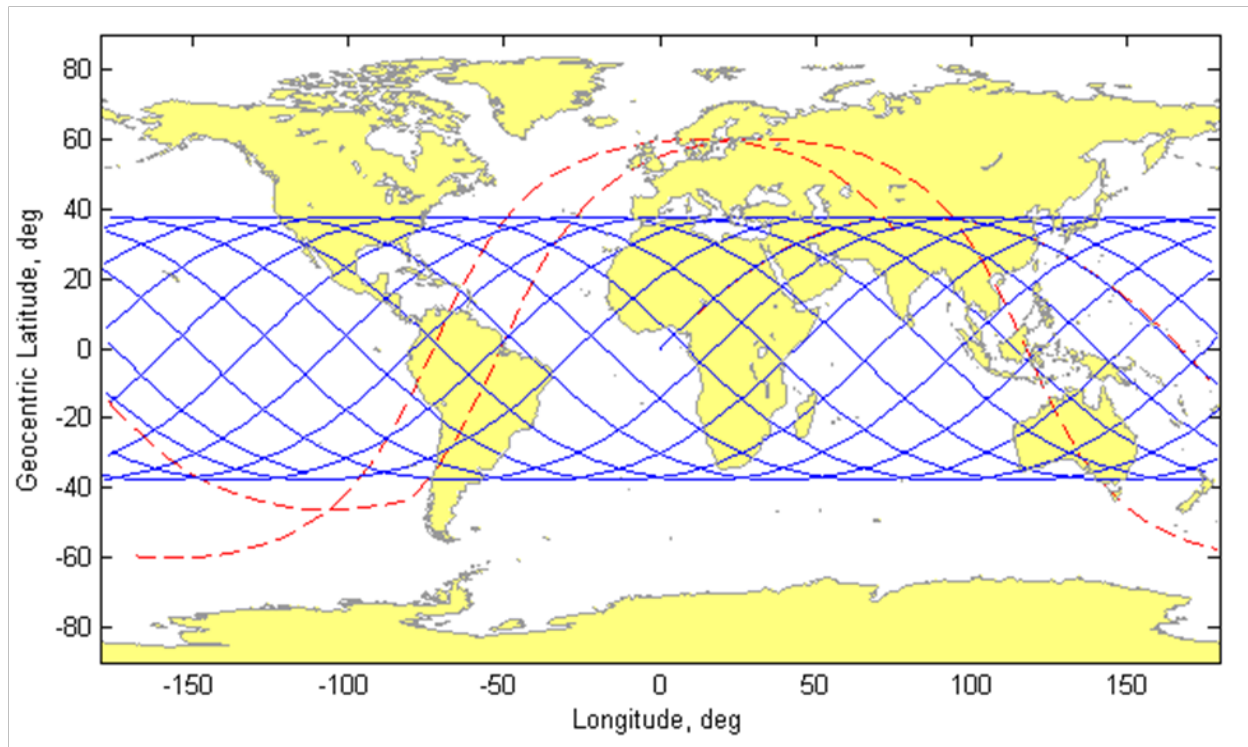


Figure 5.19. Reference Orbit (solid, blue) and Perturbed Orbit (dash, red)
Ground Track Trajectories of Single TAV Design

Plotting the altitude profile of the skip entry trajectory in Fig. 5.20 indicates that the perigee altitude of 86.75 km was reached after an elapsed time of 48.15 min, while re-circularization at skip apogee occurred after 76.67 min. As a result of losses in kinetic energy due to aerodynamic drag encountered by the TAV while transiting perigee, the skip apogee and corresponding re-circularized orbit altitude of 131.2 km is 86.88% lower than the 1000 km initial reference orbit altitude. With re-circularization near the upper limit of the sensible atmosphere, the TAV maintains a limited capability of performing subsequent maneuvers resulting from low available orbital potential energy as well as a drag-induced decaying re-circularized orbit. By performing a combined orbit-raising and re-circularization maneuver at skip apogee, a stable orbit can then be established at a higher altitude within LEO. Despite the added ΔV expenditure for such a maneuver at skip apogee, the TAV is then capable of multiple over-flights of ground targets within the $37.84 \text{ deg} \leq \phi \leq 57.75 \text{ deg}$ and $-57.75 \text{ deg} \leq \phi \leq -37.84 \text{ deg}$ latitude bands available upon completion of the orbit-raising transfer.

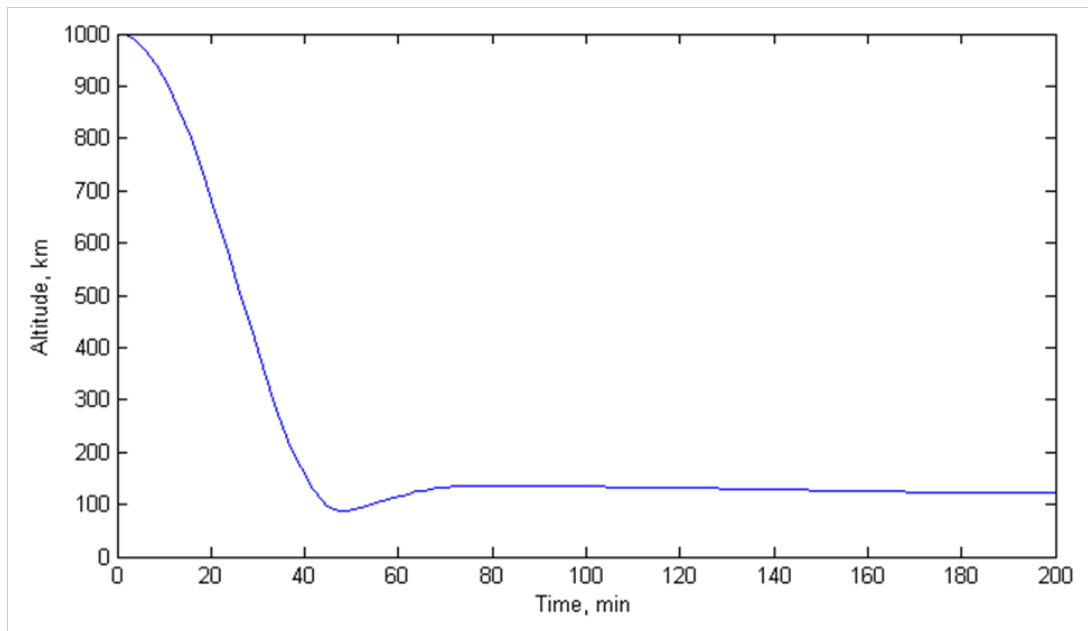


Figure 5.20. Altitude Profile for Perturbed Orbit of Single TAV Design

To ascertain the effectiveness of the skip entry maneuver with respect to ΔV expended for the inclination change achieved, a purely propulsive simple plane change maneuver performed *in vacuo* was simulated based on the equation:¹⁴⁷

$$\Delta V_{simple} = 2 {}^R V_i \cos(\gamma) \cdot \sin\left(\frac{1}{2} |\Delta i|\right) \quad (5.8)$$

The ΔV required to achieve a maximum inclination change of $\Delta i = 19.91$ deg is given in Table 5.8 with four maneuver cases: (1) Skip entry with re-circularization at skip apogee (131.2 km); (2) skip entry with re-circularization at 500 km following a Hohmann transfer performed at skip apogee; (3) skip entry with re-circularization at $h_i = 1000$ km following a Hohmann transfer performed at skip apogee; and (4) simple plane change performed at $h_i = 1000$ km.

Table 5.8. Maneuver ΔV Comparison of Orbit Re-Circularization Cases

<i>Case</i>	<i>Total ΔV</i>	<i>Percent Increase</i>
1	0.345 km/s	—
2	0.806 km/s	133.6%
3	1.068 km/s	209.6%
4	2.397 km/s	594.8%

By conducting a Hohmann transfer to boost the TAV altitude from 131.2 to 500 km at skip apogee, a ΔV increase of 133.6% is required with the total ΔV for the skip entry maneuver increasing from 0.345 km/s to 0.806 km/s. For a skip apogee boost to 1000 km, the total ΔV increases by 209.6% for the skip entry maneuver. When performed at $h_i = 1000$ km, the simple plane change requires 594.8% more ΔV than the skip entry maneuver with re-circularization at skip apogee. With re-circularization occurring at either skip apogee or a boosted altitude such as 500 km or 1000 km, the skip entry maneuver demonstrates a considerable reduction in propellant expenditure when achieving a maximum inclination change of $\Delta i = 19.91$ deg.

¹⁴⁷ Vallado, 345-346.

Summary and Conclusion

Employing the Design of Experiments method of orthogonal arrays, an optimal TAV and trajectory design can be determined for trans-atmospheric skip entry maneuvers. Satisfying the multi-objective optimization problem given in Eq. (5.1) as $\{\max(\Delta i), \min(\Delta V_{Total})\}$, the optimal solution was obtained through main effects and Pareto front analyses of the objective spaces produced by executing a series of orthogonal array experiment campaigns constructed of factors related to TAV and skip entry trajectory design. Since certain combinations of TAV and trajectory levels yielded planetary impact and mission failure, orthogonal arrays with a high density of experiments were created to improve the simulation success rate for each campaign.

Starting from a circular reference orbit with an inclination of 37.84 deg, a TAV can achieve a maximum inclination change of $\Delta i = 19.91$ deg by performing a skip entry aeroassisted maneuver with a vehicle design of $m = 2000$ kg, $S = 18.5$ m², $C_D = 0.5$, and $C_L = 3.0$, and a trajectory defined by $h_i = 1000$ km, $h_p = 86.75$ km, and $\sigma = -90$ deg. If orbit re-circularization occurs at skip apogee, then $\Delta V = 0.345$ km/s for the maneuver. With re-circularization at an altitude higher than skip apogee, such as 500 km, the total ΔV required to perform both the skip entry maneuver and Hohmann transfer is 0.806 km/s. Without an orbit-raising transfer, the preceding analysis demonstrates that a skip entry maneuver out-performs a simple plane change, with the former requiring approximately 50-85% less ΔV to achieve a maximum inclination change of $\Delta i = 19.91$ deg. Based on the vehicle and trajectory designs, the amount of inclination change achievable by a TAV is a function of the duration of atmospheric flight: longer transit-times in the atmosphere increase the exposure of a TAV to aerodynamic forces and, as a result, enhance the ability of the TAV to perform an aerodynamic turn and change orbit inclination.

VI. Low Earth Orbit Injection and Reachability Utilizing Descent-Boost Maneuvers

Chapter Overview

Similar to an aerobang trans-atmospheric maneuver, the descent-boost maneuver is introduced as an alternative to the exo-atmospheric combined Hohmann and bi-elliptic transfers for injection into a desired low Earth orbit. Utilizing a notional trans-atmospheric, lifting re-entry vehicle with $L/D = 6$, circular orbit injection simulations demonstrate that despite requiring a longer time-of-flight than bi-elliptic transfers, descent-boost maneuvers require 6-12% less ΔV for injection altitudes lower than 650 km for initial altitude cases of 1000, 1100, and 1200 km. In addition, the concept of the *Maneuver Performance Number* is introduced as a dimensionless means of comparative effectiveness analysis for both exo- and trans-atmospheric maneuvers.

Introduction

Defined as a special case of lifting entry, a descent-boost maneuver is comprised of exo- and trans-atmospheric trajectory segments as described by the example in Fig. 6.1. For the present research, a descent-boost maneuver commences with two consecutive impulses applied by the TAV at an initial circular orbit altitude, h_i (A). The first impulse (ΔV_γ), or “descent” ΔV , creates a de-orbit trajectory by altering the flight-path angle such that $\gamma_i < 0$. The second impulse (ΔV_{Boost}), or “boost” ΔV , increases the orbital velocity of the TAV. Following (A), orbital altitude decreases until perigee transit at (B), which occurs below the upper limit of the sensible atmosphere at an altitude of approximately 120 km. As the perigee altitude of a descent-boost trajectory decreases, the TAV encounters increasing atmospheric density and, therefore, greater aerodynamic drag and heating effects. With the completion of an orbit injection impulse (ΔV_{Inject}) at skip apogee (C), the TAV enters either a circular or elliptical orbit as prescribed by

mission requirements. The total ΔV required to perform a descent-boost maneuver is given by the following:

$$\Delta V_{DB} = \Delta V_{\gamma} + \Delta V_{Boost} + \Delta V_{Inject} \quad (6.1)$$

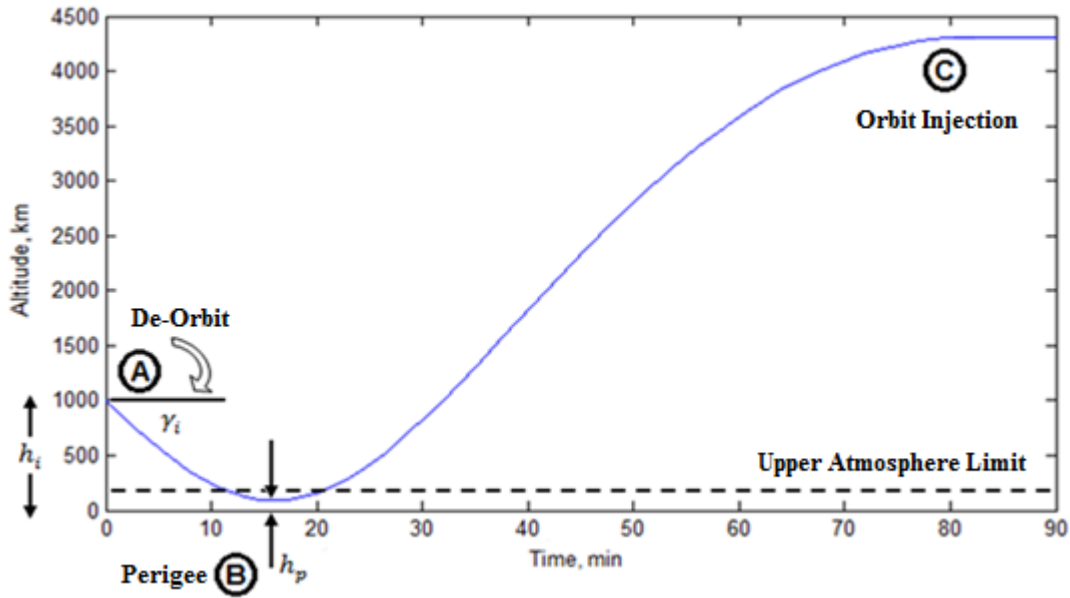


Figure 6.1. Descent-Boost Maneuver Diagram

Apart from inclination-centric orbit transfer analysis, the implementation of aeroassisted maneuvers for orbit injection via changes to semi-major axis has received minimal attention and represents the core focus of the present research of descent-boost maneuvers.

Maneuver Performance (MP) Number

Whether formulated according to physical similarity criteria or experimental results, Kuneš states that a dimensionless quantity is fundamentally comprised of either a simple ratio of two dimensionally equal quantities, or a composed ratio of dimensionally equal products of

quantities.¹⁴⁸ Despite the existence of numerous dimensionless quantities in the field of aerospace engineering, such as those related to fluid mechanics¹⁴⁹ and heat transfer,¹⁵⁰ no ratios have been devised pertaining to spacecraft maneuver effectiveness. For a given maneuver, performance can be measured in terms of several parameters of varying scale: ΔV expenditure, time-of-flight, change in orbit altitude and geometry, and change in orbital plane orientation. In an effort to reduce to the number of parameters and facilitate maneuver comparative analyses, the Maneuver Performance (MP) number is formulated as:¹⁵¹

$$\mathfrak{p} = \frac{(TOF)\Delta V}{|\Delta h| \cos \Delta i} \quad (6.2)$$

where (TOF) denotes the maneuver time-of-flight in seconds, Δh is the change in orbit altitude, or $\Delta h = h_f - h_i$, and Δi is the change in orbit inclination in radians, or $\Delta i = i_f - i_i$. A form of dimensionless cost-effectiveness ratio, the MP number represents the ratio of maneuver cost to maneuver action.¹⁵²

As examples of MP number implementation, Table 6.1 provides maneuver information related to the execution of Hohmann, one-tangent, and bi-elliptic exo-atmospheric transfers for two cases: (1) Transfer from LEO to GEO; and (2) transfer from LEO to lunar orbit. With the first case, the one-tangent burn yields the lowest MP number of $\mathfrak{p} = 1.6$ and is thus considered the most effective maneuver option. Even though the Hohmann transfer requires the least ΔV expenditure, a 6476.4 sec longer time-of-flight than the one-tangent burn produces a higher MP

¹⁴⁸ Joseph Kuneš, *Dimensionless Physical Quantities in Science and Engineering* (Waltham, MA: Elsevier Inc., 2012), 1.

¹⁴⁹ Robert A. Granger, *Fluid Mechanics* (Mineola, NY: Dover Publications, Inc., 1995), 379-384.

¹⁵⁰ E. Marín, A. Calderón, and O. Delgado-Vasallo, "Similarity Theory and Dimensionless Numbers in Heat Transfer," *European Journal of Physics* 30 (2009): 440-441.

¹⁵¹ In the Unicode® script, the symbol for MP number represents the "Latin small letter p with hook" from the "Latin Extended-B" library (Julie D. Allen, et al., *The Unicode Standard*, 587).

¹⁵² Henry M. Levin and Patrick J. McEwan, *Cost-Effectiveness Analysis*, Second Edition (Thousand Oaks, CA: Sage Publications, Inc., 2001), 133.

number at $\beta = 2.1$. With a time-of-flight nearly 535% greater than the one-tangent burn as a result of transiting an intermediate orbit apogee of 47836.00 km prior to GEO injection, the bi-elliptic transfer maintains the highest MP number at $\beta = 9.0$. Similarly, MP number analysis of the second case indicates that the one-tangent burn is again the most effective option, while the bi-elliptic transfer remains the least effective. For a ΔV savings of 4.76%, the bi-elliptic transfer requires a longer time-of-flight than the one-tangent burn at $\Delta TOF = 1839088.4 \text{ sec} = 21.2$ days, thus substantiating the higher MP number of $\beta = 22.2$.

Table 6.1. MP Number Usage Examples with Exo-Atmospheric Maneuvers¹⁵³

<i>Type</i>	$h_i, \text{ km}$	$h_f, \text{ km}$	$\Delta V, \text{ km/s}$	$TOF, \text{ sec}$	$\Delta i, \text{ deg}$	β
Hohmann	191.344	35781.35	3.935	18921.6	0.0	2.1
1-Tangent			4.699	12445.2	0.0	1.6
Bi-Elliptic	(LEO)	(GEO)	4.076	78998.4	0.0	9.0
Hohmann	191.344	376310	3.966	427258.8	0.0	4.5
1-Tangent			4.099	299019.6	0.0	3.3
Bi-Elliptic	(LEO)	(Lunar)	3.904	2138108	0.0	22.2

While applicable for both exo- and trans-atmospheric maneuvers, the MP number as expressed in Eq. (6.2) is restricted to maneuvers cases with unequal initial and final altitudes, thus precluding the analysis of phasing maneuvers. For maneuvers featuring $h_i = h_f$, the following variation can be utilized:

$$\beta_p = \frac{(TOF)\Delta V}{|\Delta h_{max}| \cos \Delta i} \quad (6.3)$$

where the subscript p indicates “phasing,” and Δh_{max} is given by $\Delta h_{max} = h_{max} - h_i$, with h_{max} representing the altitude of the greatest spatial deviation from the initial orbit altitude.

¹⁵³ Vallado, 338.

Descent-Boost Maneuver Sensitivity Study

In contrast to the skip entry maneuver which relies on changes to both bank angle and the depth of atmospheric penetration as dictated by the perigee altitude, the descent-boost maneuver instead alters the orbital trajectory of a TAV by modifying the initial flight-path angle and orbital velocity. The impact of varying these parameters on TAV trajectory geometry is explored through a sensitivity study comprising the following phases:

- (1) Commencing from a circular reference orbit as defined by Table 6.2 and the initial altitudes of $h_i = 500, 1000, 2000, 5000$ km, the initial orbital velocity is modified according to changes in the boost impulse of $\Delta V_{Boost} = 0.3, 0.5, 0.8$, and 1.0 km/s applied at $t = 0$ with a constant initial flight-path angle and a bank angle of $\sigma = 0$ deg.

Table 6.2. Reference Orbit Initial States for Descent-Boost Simulations

Eccentricity, e	0.0
Longitude, θ_i	0 deg
Latitude, ϕ_i	0 deg
Inclination, i_i	70 deg
Flight-Path Angle, γ_i	0 deg
Heading Angle, $^l\psi_i$	70 deg

- (2) Based on constant values for initial altitude and ΔV_{Boost} , the initial flight-path and inclination angles are varied within the respective intervals $\gamma_i \in [-19.5 \text{ deg}, -1 \text{ deg}]$ and $i \in [0 \text{ deg}, 80 \text{ deg}]$, with $\sigma = 0$ deg.

For each sample initial altitude within the first phase, the initial flight-path angle represents the greatest angle magnitude that does not produce a planetary impact trajectory for a descent-boost maneuver with $\Delta V_{Boost} = 0.3$ km/s. As shown in Table 6.3, the skip apogee altitude is a function of both boost impulse and perigee altitude. With ΔV_{Boost} increasing from

0.3 km/s to 1.0 km/s, the skip apogee altitude increases since the higher initial levels of orbital kinetic energy produce a shallower depth of atmospheric penetration. Accordingly, a shallower perigee altitude with respect to a given initial orbit altitude creates an increase in skip apogee altitude due to diminished aerodynamic drag losses. For $h_i = 500$ km, the increase in perigee altitude from 63 km to 214 km produces a change in skip apogee altitude of 4762 km as ΔV_{Boost} increases from 0.3 km/s to 1.0 km/s. Commencing from the higher altitude of $h_i = 5000$ km, the same increase in boost impulse produces a skip apogee change of 16929 km – an increase approximately 3.5 times greater than the $h_i = 500$ km case. When h_a vs. h_i is plotted for each boost impulse case as shown in Fig. 6.2, linear regression analysis yields a squared correlation coefficient of $R^2 = 0.9993$ for $\Delta V_{Boost} = 0.3$ km/s, $R^2 = 0.9989$ for $\Delta V_{Boost} = 0.5$ km/s, $R^2 = 0.9965$ for $\Delta V_{Boost} = 0.8$ km/s, and $R^2 = 0.9977$ for $\Delta V_{Boost} = 1.0$ km/s. Despite penetrating the deepest into the sensible atmosphere and experiencing greater nonlinear drag effects, the skip apogee altitudes reached with $\Delta V_{Boost} = 0.3$ km/s retain the strongest linear relationship with initial orbit altitude for all simulated cases.

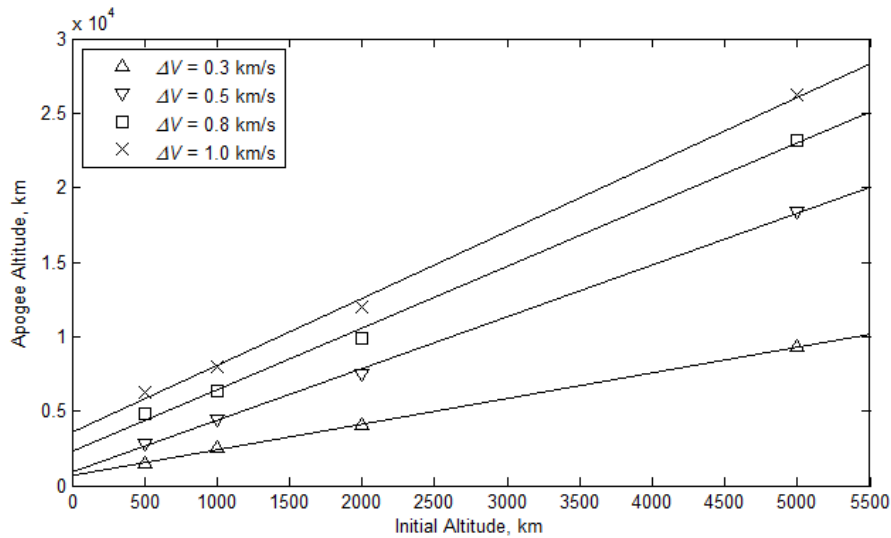


Figure 6.2. Descent-Boost Apogee Altitude with Variable Initial Altitude and Boost Impulse

In addition to skip apogee altitude, the boost impulse and perigee altitude also influence the maximum deceleration and stagnation heat flux experienced by the TAV during the trans-atmospheric segment of the descent-boost trajectory. From Table 6.3, the deceleration increases as the depth of atmospheric penetration increases, with the greatest deceleration of 63.25 g resulting from the TAV transiting the lowest perigee altitude with the highest velocity of the sample cases. As periapsis becomes shallower and higher in altitude than the upper atmosphere limit, the deceleration thus decreases as aerodynamic drag decreases. For descent-boost maneuvers featuring a perigee altitude of $h_p > 120$ km, the deceleration experienced by the TAV becomes less than unity. Adhering to the same physical trends as deceleration, stagnation heat flux reached a maximum among the sample simulations cases of 10709 kW/m² for the initial conditions $h_i = 5000$ km and $\Delta V_{Boost} = 0.3$ km/s. By comparison, the Space Shuttle mission STS-5 experienced a maximum re-entry heat flux of 1400 kW/m² on the lower surface of the wing leading-edge at an approximate altitude of 70 km.¹⁵⁴ Unlike the descent-boost case which only represents an estimate of stagnation heat flux, the STS-5 measurement is total heat flux and, therefore, includes contributions by radiative heating.

¹⁵⁴ Ko, "Finite Element," 16, 18, 32.

Table 6.3. Trajectory Parameters for Descent-Boost Maneuvers with Variable Boost ΔV at $\sigma = 0$ deg

<i>Parameter</i>	$\Delta V_{Boost}, km/s$	<i>Case</i>			
		<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>
h_i, km	—	500	1000	2000	5000
γ_i, deg	—	−8.0	−11.7	−19.0	−33.8
h_p, km	0.3	63	57	50	49
	0.5	71	133	107	158
	0.8	166	319	386	538
	1.0	214	403	532	761
h_a, km	0.3	1505	2509	4019	9308
	0.5	2776	4480	7495	18377
	0.8	4852	6334	9910	23214
	1.0	6267	7952	11991	26237
Max. Decel, g	0.3	7.186	16.07	50.11	63.25
	0.5	2.393	0.336	0.500	0.884
	0.8	0.340	0.408	0.572	0.950
	1.0	0.397	0.464	0.622	0.950
$\dot{Q}_{s,max}, kW/m^2$	0.3	2717	4316	8249	10709
	0.5	1627	19.12	82.38	13.99
	0.8	8.600	0.764	0.410	0.140
	1.0	3.511	0.330	0.126	0.038

The second phase of the sensitivity study executed a series of single descent-boost maneuvers with constant values for initial altitude and boost impulse of $h_i = 2000$ km and $\Delta V_{Boost} = 0.5$ km/s, respectively. Based on the preceding phase, these initial conditions approximate the median values of the intervals $h_i \in [500, 5000]$ km and $\Delta V_{Boost} \in [0.0, 1.0]$ km/s. As shown in Fig. 6.3(a), the ΔV required to complete a descent-boost maneuver – to include circular orbit injection at skip apogee – increases as initial flight-path angle changes from $\gamma_i = -1$ deg to $\gamma_i = -19.5$ deg since the descent impulse ΔV_γ increases as the flight-path angle increases in magnitude. In terms of orbital plane orientation, the maneuver ΔV decreases as inclination increments from 0 deg to 80 deg, thus indicating a greater propellant cost for performing a descent-boost maneuver near the equator.

Although $\sigma = 0$ deg, Fig. 6.3(b) illustrates a change in inclination angle (Δi) for all descent-boost maneuvers performed, with the magnitude of Δi related to changes in both initial flight-path angle and inclination. As the magnitude of these parameters increase, a negative inclination change is created and, consequently, a contraction of the orbit trajectory with respect to latitude. When the bank angle is changed to $\sigma = -90$ deg, however, the amount of inclination change remains relatively constant with a RMS deviation of 2.3318×10^{-4} deg for the $i = 80$ deg case with $\gamma_i \in [-18.3, -1]$ deg. For all inclination cases, the initial flight-path angles of $\gamma_i \in [-19.5, -18.3]$ deg could not be simulated since they produced planetary impact scenarios when $\sigma = -90$ deg. Limited to a single initial altitude case, a cursory assessment of Δi solution behavior indicates a strong dependence on both initial flight-path angle and inclination, and a weak dependence on bank angle for descent-boost maneuvers.

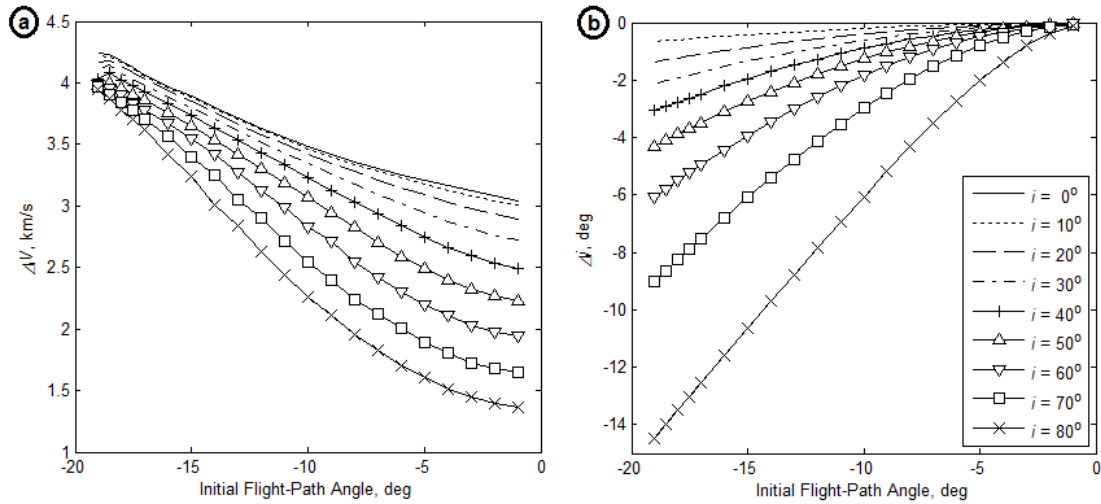


Figure 6.3. Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) ΔV vs. γ_i , and (b) Δi vs. γ_i

Figure 6.4(a) shows that as the initial flight-path angle magnitude increases, the apogee altitude generally decreases due to a deeper penetration into the sensible atmosphere by the TAV as given in Subplot (b). For initial inclinations of $i > 40$ deg, Subplot (b) indicates a shallower perigee altitude either near or higher than the upper atmosphere limit. Consequently, aerodynamic drag losses are reduced and thus greater apogee altitudes are shown in Subplot (a) as initial flight-path angle changes from $\gamma_i = -1$ deg to $\gamma_i = -19.5$ deg. As the apogee altitude increases, Fig. 6.5(a) illustrates a likewise increase in ΔV that is approximately linear in nature for each inclination case. With all descent-boost maneuvers simulated incurring an inclination change, the combined Hohmann transfer was selected as the comparative maneuver rather than the planar Hohmann or bi-elliptic alternatives since it changes both inclination and semi-major axis. Shown in Fig. 6.5(b), the combined Hohmann transfer requires less ΔV than the descent-boost maneuvers to reach apogee for all combinations of initial conditions. While the ΔV is nearly equivalent for $0 \text{ deg} \leq i < 40 \text{ deg}$, a divergence is seen for $i \geq 60 \text{ deg}$ as a result of a higher Δi produced by the descent-boost maneuvers for these initial inclination cases.

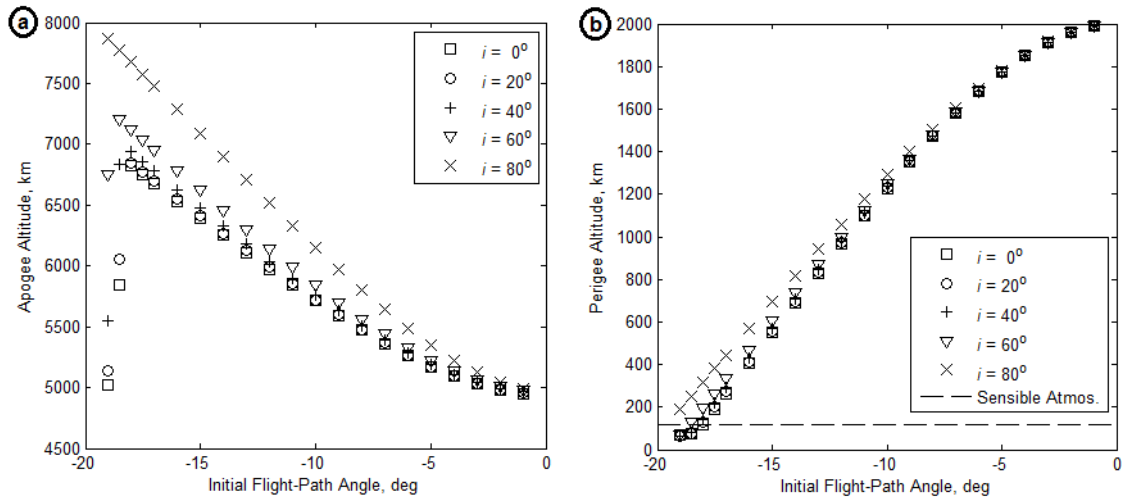


Figure 6.4. Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000 \text{ km}$, $\Delta V_{Boost} = 0.5 \text{ km/s}$, $\sigma = 0 \text{ deg}$; (a) h_a vs. γ_i , and (b) h_p vs. γ_i

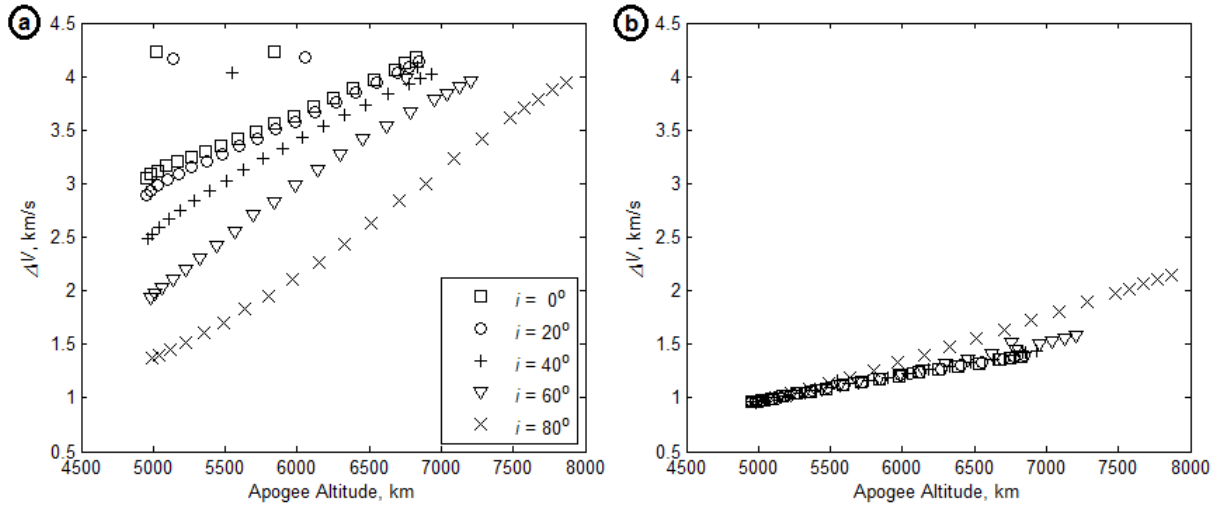


Figure 6.5. Comparison of ΔV vs. Apogee Altitude Performance with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Descent-Boost Maneuvers, and (b) Combined Hohmann Transfer Maneuvers

In an effort to reduce the number of comparative parameters between the descent-boost maneuver and combined-Hohmann transfer, MP number analysis was performed to yield surface plots as given in Figs. 6.6 and 6.7. For descent-boost maneuvers, Fig. 6.6 demonstrates that the greatest maneuver effectiveness corresponds to the global minimum of the MP number surface, where initial flight-path angle magnitude is at a minimum and initial inclination is a maximum. Conversely, descent-boost maneuvers become increasingly less effective due to higher ΔV costs as the initial flight-path angle increases in magnitude and the inclination approaches zero at the equator. Mirroring the graphical trend in Fig. 6.5(b), the MP number surface in Fig. 6.7 shows a nearly horizontal orientation with the exception of the region corresponding to both high initial flight-path angle and inclination. Requiring approximately half of the ΔV expenditure as the descent-boost maneuver, the overall magnitude of the MP number surface for the combined Hohmann transfer is likewise approximately half in magnitude.

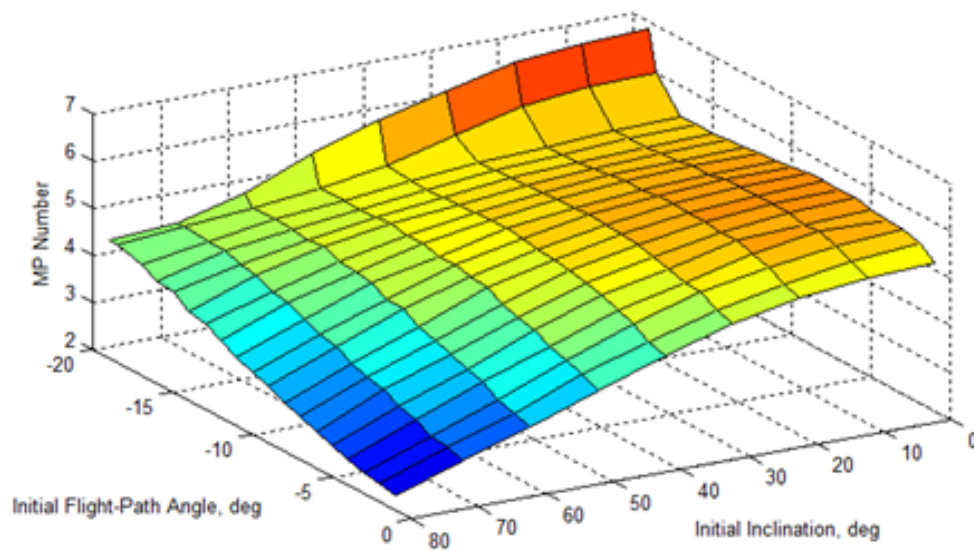


Figure 6.6. Maneuver Performance (MP) Number Analysis for Descent-Boost Maneuvers with Variable Initial Inclination, $h_i = 2000$ km, $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg

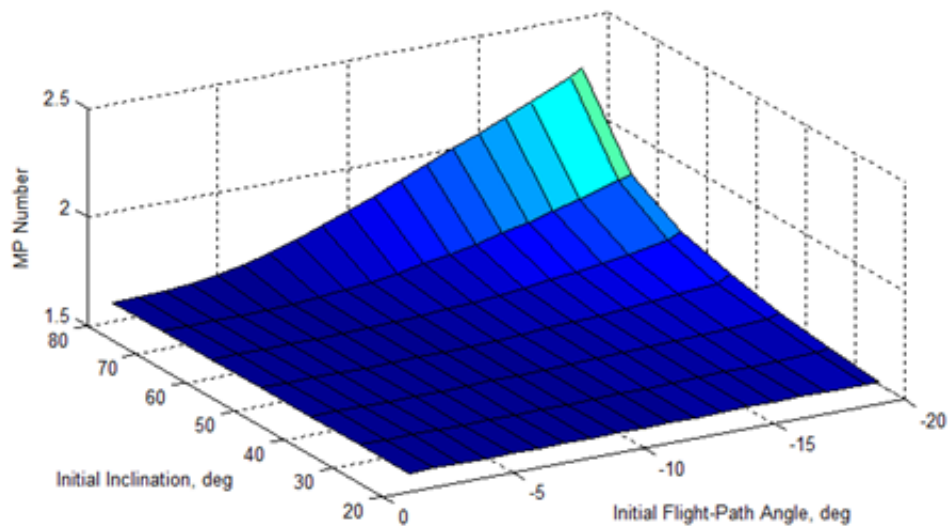


Figure 6.7. Maneuver Performance (MP) Number Analysis for Combined Hohmann Transfer Maneuvers with Variable Initial Inclination and $h_i = 2000$ km

Results and Analysis

Even though the maneuver diagram in Fig. 6.1 assumed orbit injection at skip apogee, descent-boost maneuvers are capable of performing ΔV_{Inject} impulses within the altitude range $h_{Sensible} < h_{Inject} \leq h_a$ if multiple skips in the atmosphere are permitted by mission time requirements. The upper limit of the sensible atmosphere at approximately 120 km is given as the lower bound for orbit injection altitude since the region $h_p \leq h \leq h_{Sensible}$ cannot produce a stable orbit as a result of aerodynamic drag. With orbits near $h_{Sensible}$ encountering sufficient drag forces to create a decaying trajectory, it is assumed that a subsequent orbit-raising maneuver (e.g. Hohmann transfer) will be performed if TAV mission end-of-life re-entry is not desired.

An example of an orbit injection occurring at an altitude lower than skip apogee is given in Fig. 6.8. In Subplot (a) of said figure, a descent-boost maneuver is executed from $h_i = 1000$ km and $\gamma_i = -12.5^\circ$, which places perigee at $h_p \approx 76$ km. Rather than injecting into either a circular or elliptical orbit at skip apogee, the TAV transits apogee and again reaches perigee located within the sensible atmosphere. Due to aerodynamic drag, the apogee of the elliptical orbit created by the descent-boost maneuver decays with eventual planetary impact occurring between $700 < t < 900$ min. For a non-apogee orbit injection, the target altitude of 500 km is selected and illustrated in Subplot (a). So as to minimize the total descent-boost ΔV , $\min(\Delta V_{Inject})$ is achieved by first calculating ΔV_{Inject} for each crossing of the trajectory with the target altitude, then performing a global comparison of all injection impulses to select the minimum value. From Subplot (a), $\min(\Delta V_{Inject})$ occurs after the sixth perigee passage and produces the desired orbit injection after $t \approx 550$ min in Subplot (b).

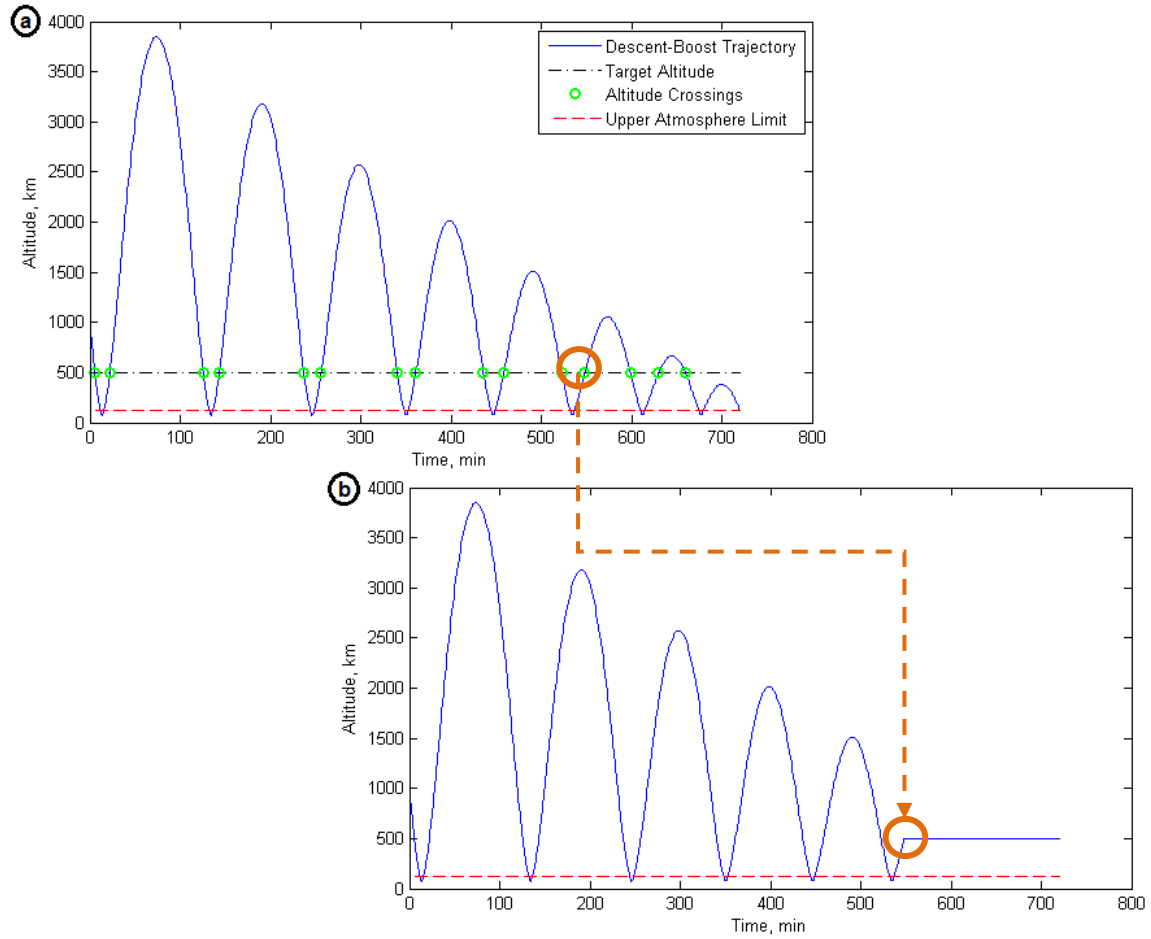


Figure 6.8. Example Circular Orbit Injection via Descent-Boost Maneuver;
(a) Truncated Descent-Boost Trajectory with Target Altitude Crossings, and
(b) Trajectory with Re-Circularization at $\min(\Delta V_{Inject})$

The circular orbit injection given in Fig. 6.8 is also shown in Fig. 6.9 as a three-dimensional polar view so as to highlight the decaying elliptical orbit of the precessing trajectory. Shifting from a trajectory color of yellow to red following the sixth perigee passage, injection occurs at an altitude of 500 km and the elliptical orbit created by the maneuver is thus re-circularized.

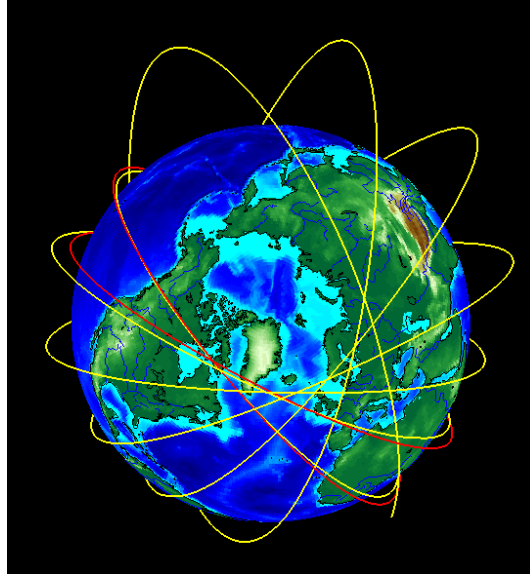


Figure 6.9. Three-Dimensional View of Descent-Boost 500 km Circular Orbit Injection with $\gamma_i = -12.5^\circ$, $\Delta V_{Boost} = 0.5 \text{ km/s}$, $h_i = 1000 \text{ km}$, $h_p \approx 76 \text{ km}$, $\sigma = 0 \text{ deg}$

Circular Orbit Injection

Outlined in Table 6.4, a series of six cases were devised to illustrate the circular orbit injection performance of descent-boost maneuvers compared with the combined Hohmann and bi-elliptic transfer alternatives performed in the vacuum environment. As a result of changes to inclination arising with the execution of the descent-boost maneuvers, the combined Hohmann transfer is utilized rather than the conventional planar Hohmann transfer since the former alters both orbital inclination and semi-major axis during the maneuver. Although planar by definition, the bi-elliptic transfer is simulated since it more closely approximates the altitude evolution of the descent-boost maneuver than the combined Hohmann. For all bi-elliptic transfers, the apogee of the intermediate orbit will equal the altitude of the first skip apogee created by the descent-boost maneuver, thereby yielding an estimate for ΔV which reflects not only orbit injection, but also the transit of the greatest altitude deviation imparted by the descent-boost maneuver.

Table 6.4. Comparison of Circular Orbit Injection Performance for Descent-Boost Maneuvers, Combined Hohmann, and Bi-Elliptic Transfers

<i>Parameter</i>	<i>Case</i>					
	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
h_i , km	1000	1000	1000	800	800	800
h_{Inject} , km	1000	1800	500	800	1800	500
γ_i , deg	-12.5	-12.5	-12.5	-10.9	-10.9	-10.9
ΔV_γ , km/s	0.9077	0.9077	0.9077	0.8356	0.8356	0.8356
ΔV_{Boost} , km/s	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
ΔV_{Inject} , km/s	0.0157	0.0069	0.0141	0.0194	0.1721	0.0112
ΔV_{DB} , km/s	1.4234	1.4146	1.4218	1.3549	1.5076	1.3468
ΔV_{Comb} , km/s	1.1068	1.2055	1.1757	1.0338	1.0641	1.0498
ΔV_{Bi-Ell} , km/s	1.1450	1.1205	1.4014	1.0721	1.0453	1.2300
TOF_{DB} , min	427.36	221.14	548.14	345.15	101.08	428.97
TOF_{Comb} , min	68.50	68.49	68.50	65.25	64.25	64.25
TOF_{Bi-Ell} , min	136.99	141.71	134.09	128.51	134.30	126.80

As a preliminary examination of descent-boost maneuver performance for orbit injection, the initial altitudes of 800 km and 1000 km were selected as well as a set of target injection altitudes located above, below, and at the same altitude as the initial condition. Similar to the sensitivity study, the initial flight-path angle selected for each case permits the deepest atmospheric penetration without planetary impact. In terms of trajectory design, $\gamma_i = -12.5$ deg produces a perigee altitude of $h_p \approx 76$ km, whereas the shallower flight-path angle of $\gamma_i = -10.9$ deg produces a perigee at $h_p \approx 75$ km due to a lower initial altitude.

With a constant ΔV_γ and ΔV_{Boost} for each initial altitude set, the variation in total maneuver ΔV arises with the selection of orbit injection altitude. For the $h_i = 1000$ km case set, the lowest injection impulse ΔV_{Inject} corresponds to a target altitude of 1800 km. In contrast, the

lowest ΔV_{Inject} for the $h_i = 800$ km case set is associated with an altitude of 500 km. By decreasing the initial altitude by 200 km, the descent-boost maneuver ΔV decreases by 4.9% for both $h_i = h_{Inject}$ and $h_i > h_{Inject}$, and increases by 7.1% for $h_i < h_{Inject}$.

When compared with the exo-atmospheric maneuvers, however, the descent-boost maneuvers maintain the highest ΔV and longest time-of-flight with the exception of the bi-elliptic transfer in Case #5. Despite featuring a time-of-flight savings of 33.22 min, the ΔV associated with the descent-boost maneuver is 1.5076 km/s – a value 44.2% greater than the bi-elliptic transfer ΔV . Overall, the combined Hohmann transfer maintains both the lowest ΔV and time-of-flight for each orbit injection case. While explicitly the superior maneuver, the combined Hohmann transfer performance is a direct function of maneuver design. Unlike the descent-boost and bi-elliptic alternatives which host at least one intermediate trajectory between the initial and target orbits, the combined Hohmann produces the most direct orbit injection scenario with the maneuver altitude restricted to either $h_{Inject} \leq h \leq h_i$ or $h_i \leq h \leq h_{Inject}$. As a consequence of not transiting the first skip apogee altitude of the descent-boost maneuver, the combined Hohmann transfer was excluded from subsequent comparative simulations in favor of the bi-elliptic transfer which provides the closest approximation of the descent-boost altitude evolution.

Starting from the reference orbit states given in Table 6.2, a series of descent-boost maneuvers and bi-elliptic transfers were simulated with $h_i = [500:100:1200]$ km. For the former maneuver type, the initial flight-path angles as given in Table 6.5 permit multiple skips without planetary impact for $t \in [0, 800]$ min as well as periapsis locations below the upper atmosphere limit.

Table 6.5. Initial Flight-Path Angles and Associated Perigee Altitudes for Descent-Boost Maneuvers

h_i, km	γ_i, deg	h_p, km
500	-7.9	79
600	-8.9	79
700	-10.0	75
800	-10.9	75
900	-11.8	74
1000	-12.5	76
1100	-13.2	77
1200	-14.0	76

With $\Delta V_{Boost} = 0.5$ km/s and $\sigma = 0$ deg, Fig. 6.10(a) illustrates the ΔV required for circular orbit injection into LEO target altitudes within the range $300 \text{ km} \leq h_{Inject} \leq h_i$. Since the descent-boost trajectory is a decaying elliptical orbit, the minimization of ΔV_{Inject} creates a sinusoidal relationship between the target altitude and total ΔV . As a substitute to the scatterplot data obtained from the maneuver simulations, Subplot (a) instead portrays the trigonometric functions of injection altitude for each initial altitude case derived via regression analysis as listed in Table 6.6. Upon examination, the the mean ΔV in each sinusoid model substantiates the general maneuver performance trend initially identified in Table 6.4: the descent-boost maneuver ΔV increases as the initial altitude increases.

When the descent-boost ΔV from Fig. 6.10(a) is compared with that for the bi-elliptic transfers in Fig. 6.10(b), regions can be demarcated where the former maneuver requires a lower ΔV for orbit injection and, therefore, represents the more viable maneuver option in terms of propellant expenditure. Shown in detail in Fig. 6.11, a lower descent-boost ΔV can be identified for $h_{Inject} < 480$ km with $h_i = 1000$ km, $h_{Inject} < 630$ km with $h_i = 1100$ km, and $h_{Inject} < 600$ km with $h_i = 1200$ km.

Table 6.6. Sinusoid Models for Descent-Boost LEO Injection Maneuvers

<i>Initial Altitude, km</i>	<i>Sinusoid Model</i>	<i>RMS Error, km/s</i>
500	$\Delta V = 0.0215 \sin(0.040537h_{Inject} - 1.86469) + 1.223$	0.00410
600	$\Delta V = 0.0210 \sin(0.040020h_{Inject} + 2.20112) + 1.257$	0.00431
700	$\Delta V = 0.0310 \sin(0.025964h_{Inject} - 1.94727) + 1.328$	0.00939
800	$\Delta V = 0.0305 \sin(0.025234h_{Inject} - 2.17010) + 1.368$	0.00583
900	$\Delta V = 0.0370 \sin(0.025751h_{Inject} - 2.08581) + 1.418$	0.01383
1000	$\Delta V = 0.0305 \sin(0.030952h_{Inject} - 2.63089) + 1.433$	0.01017
1100	$\Delta V = 0.0318 \sin(0.025751h_{Inject} + 2.08581) + 1.471$	0.01663
1200	$\Delta V = 0.0330 \sin(0.025033h_{Inject} - 2.12777) + 1.510$	0.01240

By design, an orbit injection descent-boost maneuver is comprised of an initial skip apogee which transitions into a decaying elliptical trajectory that terminates when $\min(\Delta V_{Inject})$ is satisfied. A function of γ_i and ΔV_{Boost} , the altitude of the first skip apogee ostensibly dictates not only the number of feasible elliptical orbit passages before planetary impact, but also the upper bound of possible injection orbit altitudes. Implicitly, the first skip apogee provides an opportunity for augmented mission operations. Utilizing Case #3 from Table 6.4 as an example, a TAV executing a descent-boost maneuver at $h_i = 1000$ km will reach a skip apogee of $h_a \approx 3850$ km. While the ultimate mission requirement is to inject into a circular orbit at 500 km, the TAV is capable of performing a possible orbital inspection upon transiting skip apogee.

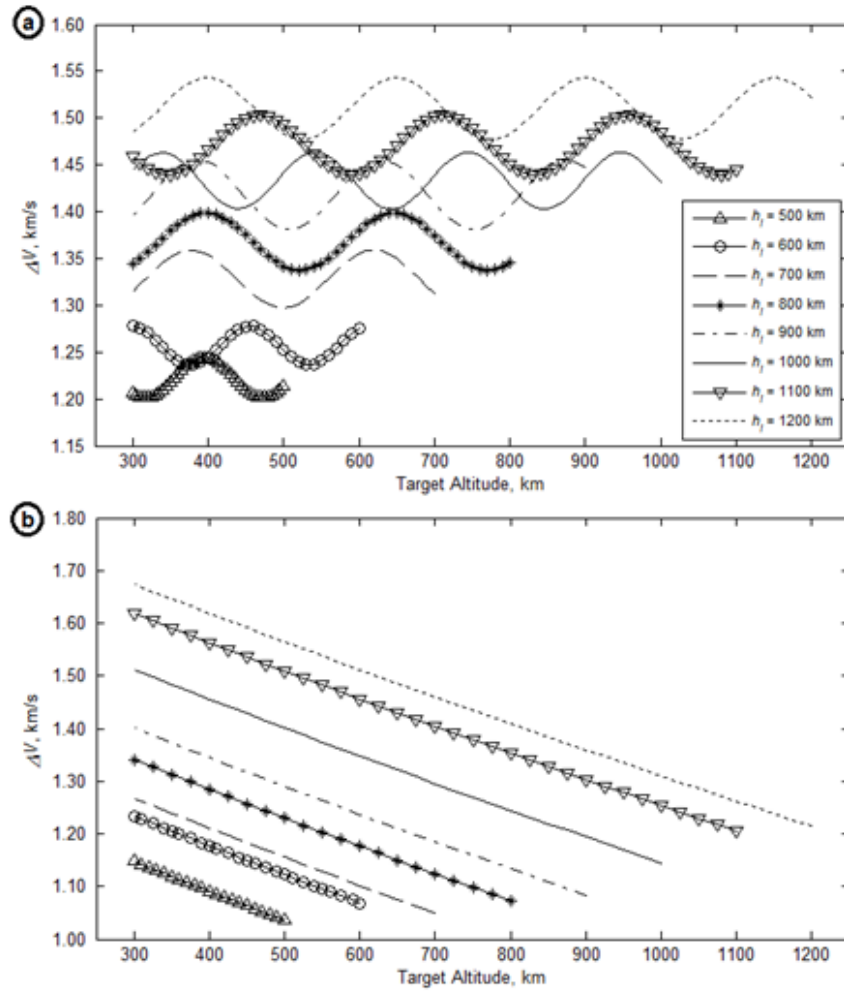


Figure 6.10. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Descent-Boost Maneuver ΔV , and (b) Bi-Elliptic Transfer ΔV

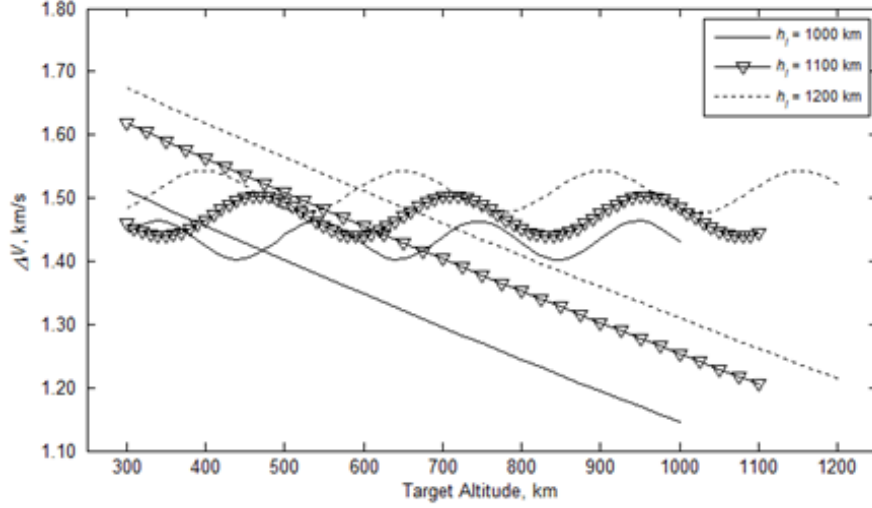


Figure 6.11. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg, and $h_i = [1000, 1100, 1200]$ km

Based on the aforementioned utility of skip apogee, Fig. 6.12 portrays the time-of-flight to reach the first apogee for both the descent-boost maneuver and bi-elliptic transfer in Subplot (a), and a quartic model for skip apogee altitude as a function of initial altitude in Subplot (b). In addition to requiring a higher orbit injection ΔV than bi-elliptic transfers – with a few cited exceptions based on the choice of both initial and injection altitudes – Subplot (a) illustrates that descent-boost maneuvers entail a longer time-of-flight to reach skip apogee. Starting at $h_i = 500$ km, the deviation in time-of-flight between the two maneuver options is $\Delta TOF \approx 4$ min; increasing the initial altitude to $h_i = 1200$ km, the deviation increases to $\Delta TOF \approx 15$ min. Pertaining only to descent-boost maneuvers, Subplot (b) depicts a regression-derived quartic model for first skip apogee altitude as described by Eq. (6.4):

$$h_a = f(h_i) = a_4 h_i^4 + a_3 h_i^3 + a_2 h_i^2 + a_1 h_i + a_0 \quad (6.4)$$

where

<i>Coefficient</i>	<i>Value</i>
a_0	-9.161075×10^3
a_1	6.078239×10^1
a_2	-1.140242×10^{-1}
a_3	9.422693×10^{-5}
a_4	-2.797119×10^{-8}

A single variable polynomial with $R^2 = 0.9989$, Eq. (6.4) assumes an average perigee altitude of $\bar{h}_p \approx 76$ km and is continuous within the interval $h_i \in [500, 1200]$ km.

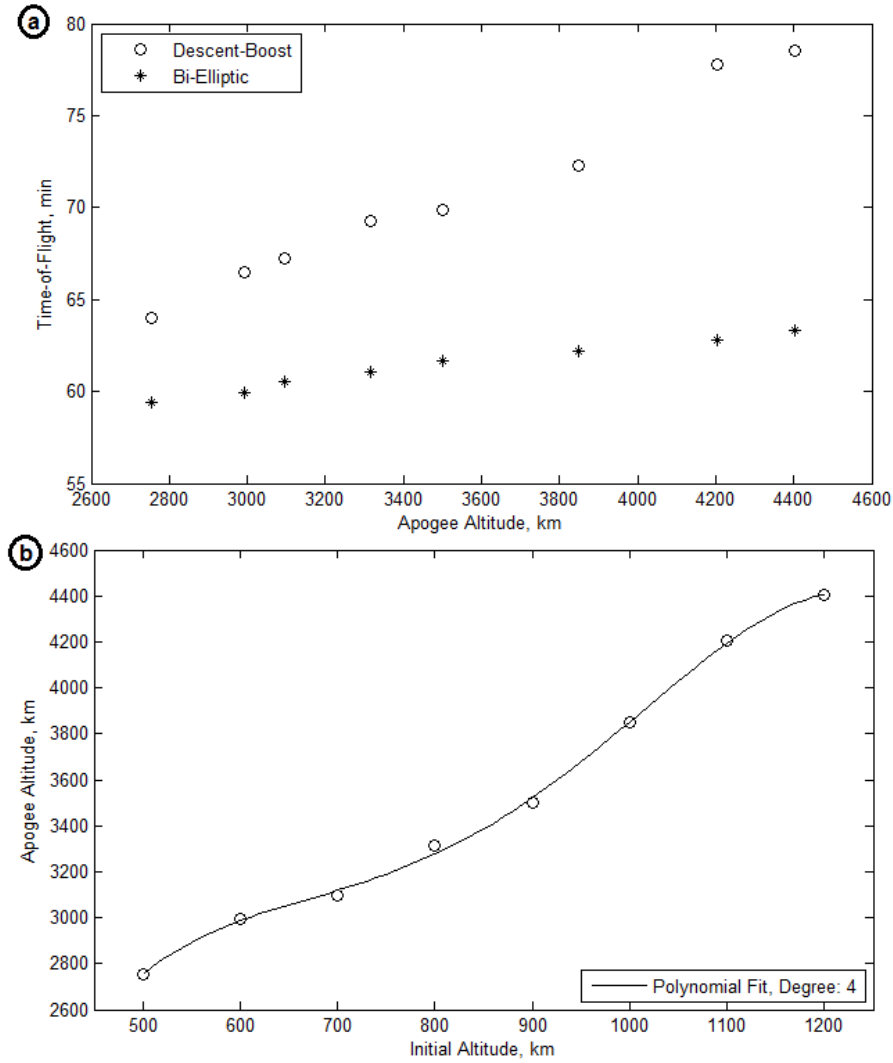


Figure 6.12. Comparison of Descent-Boost Maneuver and Bi-Elliptic Transfer with Variable γ_i , $\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg; (a) Time-of-Flight to Apogee, and (b) h_a vs. h_i for Descent-Boost Maneuvers (Quartic Model, $R^2 = 0.9989$)

Molniya Orbit Injection

The altitude reachability of descent-boost maneuvers for orbit injection is dependent on the magnitude of both the descent and boost impulses applied at $t = 0$, as well as the initial orbit altitude. With available propellant onboard the TAV representing the fundamental limiting factor, the reachability envelope becomes constrained by not only the requirements of an immediate mission tasking, but also the prospect of continued on-orbit operations. As a consequence of minimizing total ΔV expenditure, the utilization of descent-boost maneuvers for orbit injection limits the feasible reachability envelope to LEO and the transition region between LEO and medium Earth orbit (MEO), specifically $2000 \leq h_{Inject} < 5000$ km.¹⁵⁵ Although precluding injection into MEO trajectories with 12 hr periods such as those associated with the Global Positioning System (GPS) constellation, descent-boost maneuvers proffer the ability for injection into Molniya orbits. Highly elliptical orbits with eccentricities greater than 0.7 and a period approximately equal to half of one sidereal day, Molniya orbits feature a periapsis within the LEO altitude regime.¹⁵⁶

Based on the two-line element (TLE) set for the Molniya 3-42 communications satellite, an example Molniya injection orbit can be defined by a perigee and apogee altitude of 501.1350 km and $h_a = 36621.9905$ km, respectively, with an orbit inclination of 62.8 deg.¹⁵⁷ Commencing from $h_i = 1000$ km, the initial latitude/longitude coordinates $(\theta_i, \phi_i) = (0,0)$ deg, and $i = 62.8$ deg, Molniya orbit injection performance is given in Table 6.7 for the descent-boost maneuver as well as the bi-elliptic and combined Hohmann transfers. Depicted in Fig.

¹⁵⁵ I. H. Ph. Diederiks-Verschoor and V. Kopal, *An Introduction to Space Law*, Third Edition (Alphen aan den Rijn, The Netherlands: Kluwer Law International, 2008), 20.

¹⁵⁶ Charles D. Brown, *Elements of Spacecraft Design* (Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2002), 109.

¹⁵⁷ The Center for Space Standards & Innovation, "Molniya 3-42 TLE," NORAD Two-Line Element Sets, CelesTrak, last modified 29 January 2014, accessed 29 January 2014, <http://www.celstrak.com/NORAD/elements/molniya.txt>; See Appendix C for a guide to convert TLE data into Keplerian elements.

6.13, the descent-boost maneuver completes the injection impulse between the first skip apogee and the second perigee passage. Similar to previous simulations, the apogee of the bi-elliptic intermediate transfer orbit is equal to the altitude of the first skip apogee, which, for the Molniya orbit injection example is $h_a = 3906$ km. As alternatives to the descent-boost maneuver and bi-elliptic transfer, two variations of the combined Hohmann transfer are simulated: (1) Transfer from $h_i = 1000$ km to the Molniya orbit periapsis; and (2) Transfer from $h_i = 1000$ km to the Molniya orbit apoapsis.

Table 6.7. Comparison of Molniya Orbit Injection Performance for Descent-Boost Maneuver ($\Delta V_{Boost} = 0.5$ km/s, $\sigma = 0$ deg), Bi-Elliptic, and Combined Hohmann Transfer

<i>Parameter</i>	<i>Descent-Boost</i>	<i>Bi-Elliptic</i>	<i>Combined Hohmann</i>	
			<i>Perigee Transfer</i>	<i>Apogee Transfer</i>
h_i , km	1000	1000	1000	1000
h_{Inject} , km	501.1350	501.1350	501.1350	36621.9905
γ_i , deg	-12.3	0.0	0.0	0.0
ΔV_γ , km/s	0.8887	—	—	—
ΔV_{Inject} , km/s	2.0231	—	—	—
ΔV_{Total} , km/s	3.4118	2.4090	2.4301	2.3517
<i>TOF</i> , min	127.52	134.75	49.92	331.55
MP Number, β	52.3	39.0	14.6	1.3

Despite maintaining the lowest total ΔV of the maneuvers simulated, the combined Hohmann apogee transfer requires the longest time-of-flight at 331.55 min. As a result of featuring the most direct transfer trajectory between h_i and the target orbit, the combined Hohmann perigee transfer requires the shortest time-of-flight, with a savings of 281.63 min for a 7.8% increase in ΔV when compared with the apogee transfer. While representing the highest ΔV expenditure for Molniya orbit injection, the descent-boost maneuver maintains a lower time-of-

flight than both the bi-elliptic and apogee transfers. In terms of MP number, the apogee transfer is cast as the most effective maneuver since the greatest spatial distance is traversed for the lowest ΔV even though the longest time-of-flight is required. For the perigee injection cases, the combined Hohmann perigee transfer is the more effective maneuver option based primarily on a 60.9% and 63% lower time-of-flight than the descent-boost and bi-elliptic alternatives, respectively.

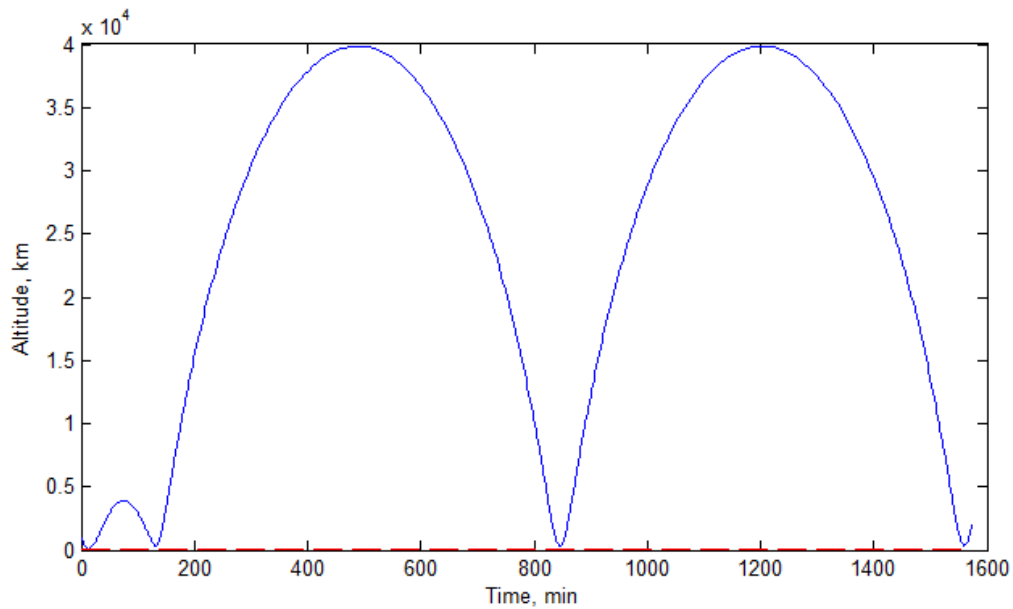


Figure 6.13. Descent-Boost Maneuver with Molniya Orbit Injection with $\gamma_i = -12.3^\circ$, $\Delta V_{Boost} = 0.5 \text{ km/s}$, $h_i = 1000 \text{ km}$, $h_p \approx 78 \text{ km}$, $\sigma = 0 \text{ deg}$

When viewed with respect to the Earth, the orbit injection scenario as shown in Fig. 6.14 reveals several details unavailable in the preceding figure, to include the first perigee passage occurring over northern Asia, skip apogee located over the South Pacific, and the signature “figure-8” geometry of the Molniya orbit.

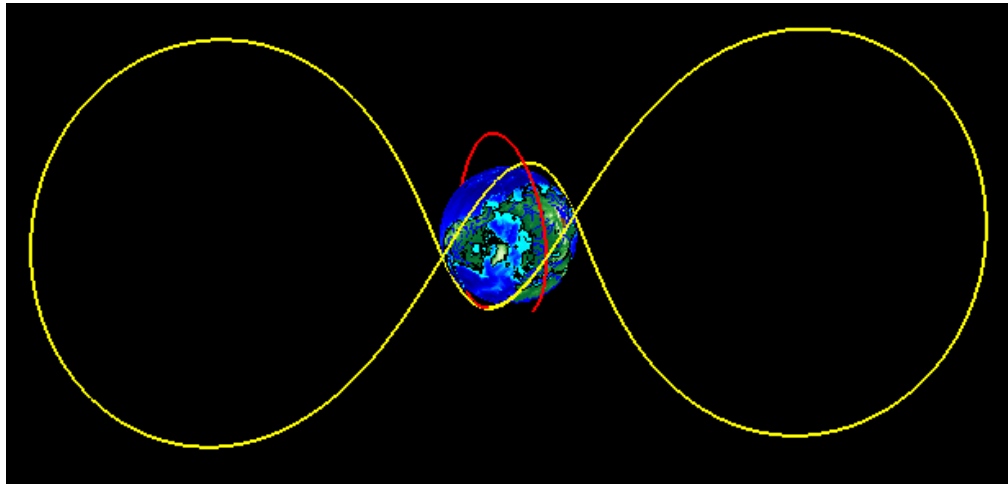


Figure 6.14. Three-Dimensional Polar View of Descent-Boost Molniya Orbit Injection

In order to reduce the total ΔV expenditure, the descent-boost maneuver was initiated without any preliminary phasing maneuver to ensure alignment with the Molniya 3-42 orbit in terms of RAAN and argument of perigee. Consequently, the Molniya injection and Molniya 3-42 trajectories share the same geometric shape, but not the same orbital orientation with respect to the Earth as shown in Fig. 6.15. Limited to a simulation time duration of 1600 min, both the Molniya injection (yellow) and Molniya 3-42 (green) trajectories only represent two complete orbital revolutions. If the simulation time were to be extended, precession effects would become evident since the Earth is modeled as a rotating central body. Even though the two trajectories do not intersect within the interval $0 \leq t \leq 1600$ min, several opportunities for possible orbit inspection exist during periods of trajectory close-approach. Located on the right-side of Fig. 6.16, the apparent point of orbit intersection corresponds to the closest approach of the two trajectories for $0 \leq t \leq 1600$ min. Employing the Hausdorff distance formula, the close-approach can be characterized as a trajectory separation distance of approximately 492 km.

Given two sets of finite points $A = \{a_1, a_2, \dots, a_m\}$ and $B = \{b_1, b_2, \dots, b_n\}$, the Hausdorff distance is defined by:¹⁵⁸

$$H(A, B) = \max(h(A, B), h(B, A)) \quad (6.5)$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} d_E(a, b) \quad (6.6)$$

From Eq. (6.6), the term $d_E(a, b)$ represents the Euclidean norm of the points between sets A and B .¹⁵⁹ In terms of time-of-flight, the close-approach occurs after an elapsed time of 375.5 min for the Molniya injection trajectory, to include the initial descent-boost maneuver and the Molniya perigee injection impulse. Also starting from $t = 0$, the close-approach for the Molniya 3-42 trajectory occurs after 369.5 min.

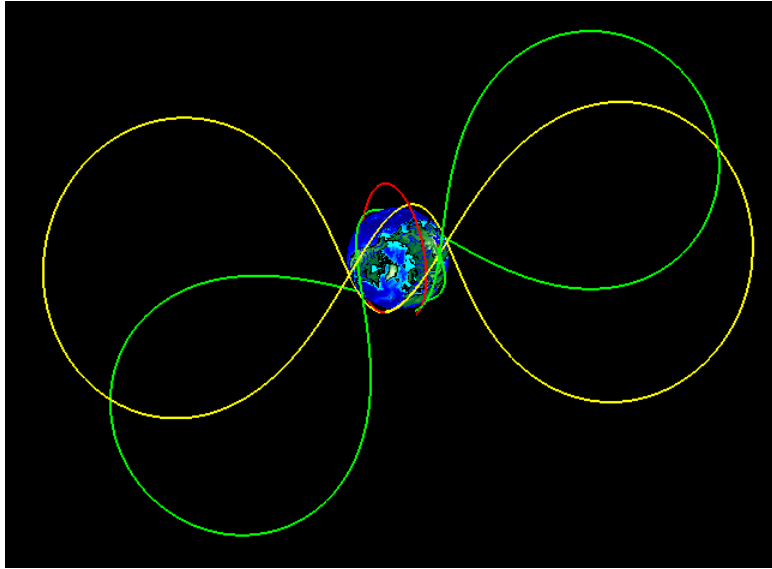


Figure 6.15. Three-Dimensional Polar View of Descent-Boost Orbit Injection and Molniya 3-42 Orbit Trajectories

¹⁵⁸ Yalin Wang, Qilong Han, and Haiwei Pan, "A Clustering Scheme for Trajectories in Road Networks," in *Advanced Technology in Teaching – Proceedings of the 2009 3rd International Conference on Teaching and Computational Science*, ed. Yanwen Wu (Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2012), 14; Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge, "Comparing Images Using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, no. 9 (1993): 850.

¹⁵⁹ Michel M. Deza and Elena Deza, *Encyclopedia of Distances*, Second Edition (Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2013), 323.

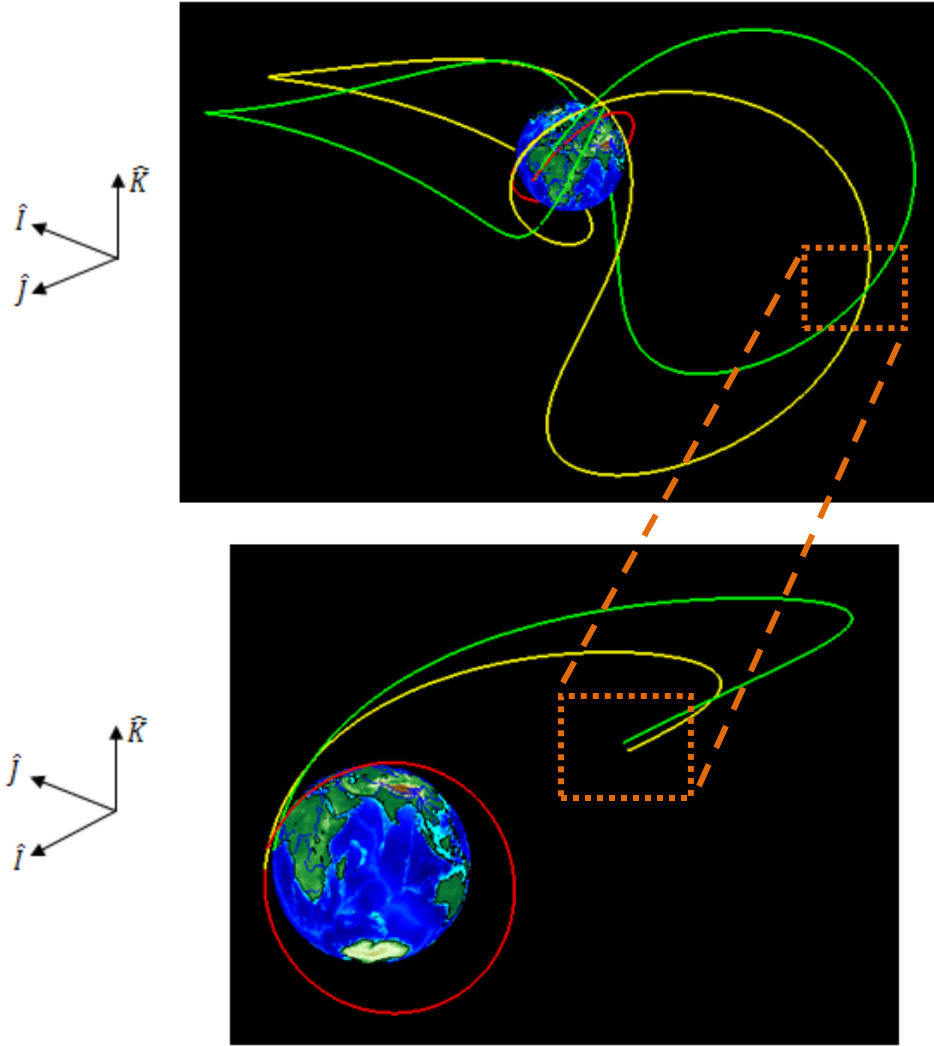


Figure 6.16. Detail of Close-Approach of Descent-Boost Orbit Injection and Molniya 3-42 Orbit Trajectories

Summary and Conclusion

A series of descent-boost maneuvers were executed to investigate maneuver performance sensitivity and the prospect of LEO injection for a notional trans-atmospheric, lifting re-entry vehicle with $L/D = 6$. When initial altitude and boost impulse are constant, simulations indicate that the total descent-boost maneuver ΔV is a strong function of initial flight-path angle and inclination, with ΔV increasing as the magnitude of both of these parameters increases. Based on

the design of the descent-boost maneuver, the requirement for an initial ΔV impulse to alter trajectory flight-path angle and orbital velocity produces an approximately 1.5 to 2.0 times greater ΔV expenditure when compared with the exo-atmospheric combined Hohmann transfer. Although requiring a longer time-of-flight for orbit injection than bi-elliptic transfers, descent-boost maneuvers are shown here to require 6-12% less ΔV for injection altitudes lower than 650 km for circular orbits with an initial altitude of 1000, 1100, and 1200 km. While MP number analysis casts the combined Hohmann transfer as the more effective option for orbit injection in terms of both ΔV expenditure and time-of-flight, descent-boost maneuvers provide two capabilities not available for the Hohmann. First, the TAV can perform an orbital inspection upon transiting skip apogee when conducting a descent-boost orbit injection; second, a degree of maneuver unpredictability is garnered since the descent-boost trajectories are trans-atmospheric by design and maintain a decaying elliptical flight-path which produces multiple orbit injection possibilities.

VII. Aeroassisted Maneuvers: Potential Air and Space Law Challenges

Chapter Overview

Diverging from the paradigm that spacecraft exclusively operate within the vacuum of outer space, current engineering efforts are seeking to create vehicles that can exploit the aerodynamic forces of the upper atmosphere in order to implement an aeroassisted, trans-atmospheric maneuver. By transcending both the airspace and outer space environments, aeroassisted maneuver represent a source of potential air and space law challenges arising due to current ambiguities surrounding the atmospheric delimitation debate as well as the question of airspace sovereignty limits vis-à-vis space law.

Introduction

Spacecraft can be divided functionally into two categories: (1) Vehicles that operate exclusively in the vacuum environment of space; and (2) vehicles that are hybrid in nature and capable of re-entering the Earth's atmosphere following the completion of a given on-orbit mission. While the former category is referred to as satellites, the latter are identified trans-atmospheric vehicles, or TAVs. Since the 1970s, TAVs have been limited to boost-glider designs, such as the Space Shuttle and X-37B Orbital Test Vehicle (OTV), which achieve orbit as either a secondary or tertiary stage on a rocket-propelled spacelift system. Following re-entry, a TAV then utilizes aerodynamic lift to perform a gliding maneuver in order to land. Seeking to evolve the boost-glider design concept, the early 21st century has witnessed an emergence of both national and corporate efforts to create a hypersonic spaceplane capable of taking off and landing horizontally on a conventional runway.¹⁶⁰ Designed as a single-stage-to-orbit vehicle,

¹⁶⁰ Carl Q. Christol, "The Aerospace Plane: Its' Legal and Political Future," *Space Policy* 9, no. 1 (1993): 36.

the spaceplane is able to attain a level of mission “availability and flexibility of use” hitherto limited by mass budget, launch window, and launch site location restrictions inherent in existing rocket booster systems.¹⁶¹

Whether designed as a boost-glider or spaceplane, TAVs offer the capability of utilizing the upper atmosphere as an alternative maneuver environment rather than an interface solely for the purpose of re-entry at the mission end-of-life for manned and unmanned spacecraft. Traditionally, orbital states and orbit geometry are modified via various maneuvers performed *in vacuo* which, depending on both the initial mission altitude and desired orbital change, have the propensity of becoming prohibitively expensive in terms of propellant expenditure. Alternatively, atmospheric re-entry can be employed as a means of operational maneuver whereby the aerodynamic drag of the upper atmosphere is exploited by a TAV to create an aeroassisted maneuver. Such maneuvers have been analytically demonstrated to achieve a desired orbital change for less propellant than required by an exo-atmospheric maneuver, thus extending the spacecraft mission life.

Applicability of Air and Space Law

Not officially defined by international treaty, the demarcation between airspace and outer space has created an extant legal debate concerning where air law ends and space law begins. As codified in Article I of the Convention on International Civil Aviation, Chicago, 1944, air law grants each state “sovereignty and exclusive territorial jurisdiction” over its’ respective airspace, only to be infringed upon by prior formal agreement or treaty.¹⁶² By contrast, Article II of the Outer Space Treaty of 1967 declares outer space to be an international zone outside the realm of

¹⁶¹ Pierre Betin, “Reflections on the Spaceplane,” *Space Policy* 7, no. 2 (1991): 138.

¹⁶² Alexandra Harris and Ray Harris, “The Need for Air Space and Outer Space Demarcation,” *Space Policy* 22, no. 1 (2006): 4; Stephan Hobe, Gerardine M. Goh, and Julia Neumann, “Space Tourism Activities – Emerging Challenges to Air and Space Law?” *Journal of Space Law* 33, no. 2 (2007): 361.

state sovereignty.¹⁶³ In order to define the transition between air and space law, two differing methods of airspace and outer space delimitation have arisen: spatialism and functionalism. For spatialists, the boundary between air and space is defined physically in terms of altitude, such as the von Kármán Line devised in the 1950s. Rather than an altitude boundary, the functionalist approach seeks to delimit airspace and outer space according to the function and “distinctive traits” of the vehicle operating within the environment in question.¹⁶⁴

Compounding the demarcation debate is the absence of any conventional or customary rule of international law addressing the “innocent passage” of vehicles ascending to or descending from space. While such passages do occur within foreign airspace, “no protests against [them] have been raised so far” and the passages are viewed as a *fait accompli*.¹⁶⁵ When considered within the context of the demarcation and atmospheric passage debates, aeroassisted maneuvers pose a series of challenges to air and space law alike. Since aeroassisted maneuvers are initiated from and terminate in space, are they governed by air and/or space law? Can such maneuvers be considered an “innocent passage” when implemented within foreign airspace?

Spatialism and Aeroassisted Maneuver Altitude Delimitation

Whether an aeroassisted maneuver is implemented to modify an existing orbit or conduct an orbital transfer from high Earth orbit to LEO, the trajectory must transit the upper atmosphere at an altitude with sufficient density to impart the requisite aerodynamic force on a TAV. Depending on the desired final orbit geometry and the imposition of deceleration and heat flux constraints by the TAV structure and/or payload, trajectory simulations have indicated that

¹⁶³ Harris, 4.

¹⁶⁴ Hobe, 363.

¹⁶⁵ Andrei D. Terekhov, “Passage of Space Objects through Foreign Airspace: International Custom?” *Journal of Space Law* 25, no. 1 (1997): 6, 8.

aeroassisted maneuvers have the potential of occurring within the 50 – 100 km altitude regime.¹⁶⁶

When viewed within the context of spatialism, the potential altitude regime for aeroassisted maneuvers occurs lower than many international attempts to delimit airspace and outer space. Corresponding to an altitude of 100 km, the von Kármán Line represents an approximate boundary above which an aircraft cannot derive any aerodynamic lift from the atmosphere and must travel at a speed approaching orbital velocity.¹⁶⁷ As an alternative, Italy in 1975 proposed a boundary at 90 km since it represented the median altitude between the upper limit of aircraft flight at 60 km, and the lowest possible satellite orbit at 120 km. In 1976, Belgium echoed the reasoning of von Kármán by advocating a boundary at 100 km, while the Soviet Union in 1979 proposed an arbitrary boundary “at an altitude not exceeding 110 km.”¹⁶⁸

Although the Italian delimitation proposal places the upper altitude limit of aircraft flight at 60 km, this corresponds to the approximate operating altitude of the X-15, an experimental rocket-propelled aircraft of the early 1960s.¹⁶⁹ In terms of conventional aircraft, the upper altitude limit is considerably lower with the U.S. Air Force’s U-2 reaching a maximum ceiling of approximately 21 km. As for spacecraft, the lowest operational orbit corresponds to an altitude of 96 km, which is lower than all aforementioned delimitation proposals.¹⁷⁰ By considering both the nominal ceiling of the X-15 and the lowest achieved satellite orbit, an altitude “gap” at

¹⁶⁶ Darby and Rao, “Minimum-Fuel,” 618-628; Patrick R. Jolley and Stephen A. Whitmore, “Aerodynamic and Propulsion Assisted Maneuvering for Orbital Transfer Vehicles” (paper presented at the 5th Responsive Space Conference, Los Angeles, CA, 23-26 April 2007): 1-39.

¹⁶⁷ Francis Lyall and Paul B. Larsen, *Space Law: A Treatise* (Surrey, United Kingdom: Ashgate Publishing Limited, 2009), 167-169.

¹⁶⁸ *Ibid.*, 169.

¹⁶⁹ W. D. Kay, “The X-15 Hypersonic Flight Research Program: Politics and Permutations at NASA,” in *From Engineering Science to Big Science: The NACA and NASA Collier Trophy Research Project Winners*, ed. Pamela E. Mack (Washington D.C.: U.S. Government Printing Office, 1998), 155.

¹⁷⁰ Katherine M. Gorove, “Delimitation of Outer Space and the Aerospace Object – Where is the Law?” *Journal of Space Law* 28, no. 1 (2000): 12.

60 – 96 km is created which aligns with the potential operating environment for aeroassisted maneuvers.

Even though conventional aircraft and lighter-than-air vehicles such as blimps do not operate at 60 – 96 km, this altitude regime is still considered to be sovereign airspace as evidenced in several reports issued since the 1960s. As one example, the Canadian government identified that the Space Shuttle *Challenger* flew within its airspace at an altitude of approximately 68 km while on glide path to land in the United States following re-entry in 1984.¹⁷¹ A second example arises from the proceedings of the United Nations' Committee on the Peaceful Uses of Outer Space in 1996.¹⁷² In response to a questionnaire disseminated to member states regarding legal issues associated with “aerospace objects,” Germany noted that the Soviet Space Shuttle *Buran* passed through the airspace of Turkey following re-entry in 1988.¹⁷³ Based on a similar structural design and mission profile, the *Buran* is assumed to have flown through Turkish airspace at an altitude commensurate with that of the *Challenger*.

Despite the ambiguity surrounding the actual spatial delimitation of airspace and outer space, precedence dictates that airspace sovereignty extends up to and beyond an altitude of 90 km. Consequently, aeroassisted maneuvers occurring at an altitude 50 – 90 km would be considered a passage through foreign airspace if not implemented over international waters.¹⁷⁴ By implementing an aeroassisted maneuver within airspace, a TAV is then subject to the jurisdiction of air law. Since an aeroassisted maneuver places a TAV within foreign airspace for a time of finite duration, however, can the passage and resulting airspace infringement be

¹⁷¹ Terekhov, 3.

¹⁷² For the complete questionnaire, see Gorove, 17-18; for the complete member states responses to the questionnaire, see . A/AC.105/635, 15 February 1996.

¹⁷³ UN doc. A/AC.105/635, 15 February 1996, at 7.

¹⁷⁴ Since aeroassisted maneuvers implemented over international waters represent benign occurrences, only those maneuvers that infringe on foreign airspace will be considered henceforth.

deemed “innocent” and thus overlooked by the overflowed state as with cases of vehicles ascending to or descending from space? The presumptive answer would be in the affirmative, but a functional analysis of the TAV mission is required in order properly classify an aeroassisted maneuver as an “innocent passage” or not.

Functionalism and TAV Classification

According to the functionalist approach, the question of legal jurisdiction is dependent on the function of the vehicle in question. Outlined in the 1975 Convention on Registration of Objects Launched into Outer Space, a launch vehicle and satellite payload are considered “space objects” and, therefore, governed by space law since they are intended to reach and operate within the space environment.¹⁷⁵ A broad term, “launch vehicle” within the context of the aforementioned Convention applies to rocket boosters and not carrier aircraft. For the latter case, such as the Lockheed L-1011 transport aircraft utilized as an upper atmospheric launching platform for the Pegasus booster rocket, both the carrier aircraft and attached spacecraft are governed by air law until vehicle separation. Following separation, the Pegasus booster and similar spacecraft cannot “derive support in the atmosphere from the reactions of the air” and are thus considered “space objects” subject to space law.¹⁷⁶

Based on the Convention on Registration, a TAV conducting normal mission operations in orbit is considered a “space object” and is subject to space law. When conducting an aeroassisted maneuver, however, the TAV utilizes aerodynamic forces within the upper atmosphere to produce lift. Does this ability to leverage aerodynamic forces during the

¹⁷⁵ *Convention on Registration of Objects Launched into Outer Space*, 14 January 1975, 28 U.S.T. 695, 1023 U.N.T.S. 15.

¹⁷⁶ Jane Van Nimmen, Leonard C. Bruno, and Linda N. Ezell, *NASA Historical Data Book, Volume VII: NASA Launch Systems, Space Transportation, Human Spaceflight, and Space Science, 1989-1998* (Washington D.C.: U.S. Government Printing Office, 1999), 55; Hobe, 364.

aeroassisted maneuver necessitate a change in vehicle status from space object to aircraft, and a likewise change in legal jurisdiction from space law to air law? Since the TAV produces lift while transiting the upper atmosphere, then it could be assumed that air law supersedes any space law consideration as with the preceding example of the Pegasus booster attached to the L-1011 carrier aircraft. The validity of this assumption is tenuous, especially when the functions of both the TAV and aeroassisted maneuver are considered. With the former, a TAV is intended to reach and operate within outer space and thus constitutes the baseline definition of a “space object.” For the latter, an aeroassisted maneuver is implemented in order to alter the geometry of an orbit, whether originally in LEO or high Earth orbit. As a result, the TAV always remains within the space environment except for the duration of the aeroassisted maneuver itself (and the eventual re-entry at mission end-of-life).

If a TAV is subject to air law during an aeroassisted maneuver, then the right of foreign airspace sovereignty must be observed. Consequently, a state whose airspace will be infringed by an aeroassisted maneuver maintains the right to regulate passage within its airspace. Apart from civilian missions such as those related to science or transportation, as with the case of space tourism, TAVs also have the potential of hosting a variety of military functions. From being a platform for augmented command, control, communications, intelligence, surveillance, and reconnaissance (C3ISR), to a vehicle for prompt global strike, TAVs proffer an undeniable enhancement of military capabilities.¹⁷⁷ Based on the their inherent military mission implications, aeroassisted maneuvers could be implemented to either deliver a TAV over a target of interest, or place a TAV inside the atmosphere to conduct a specific mission within the airspace of a state being overflown.

¹⁷⁷ Jinyuan Su, “Near Space as a *Sui Generis* Zone: A Tri-Layer Approach of Delimitation,” *Space Policy* 29, no. 2 (2013): 91.

In light of these potential missions, a state could follow precedence and impose a no-fly zone for aeroassisted maneuvers deemed to fall outside the bounds of an “innocent passage.” For example, the French and Spanish governments imposed no-fly zones which prevented the passage of U.S. Air Force aircraft through their respective airspaces when executing Operation El Dorado Canyon against Libya in 1986.¹⁷⁸ Similarly, a state could impose a no-fly zone precluding an aeroassisted maneuver intended to insert a TAV in orbital position to complete a specific military mission, e.g. C3ISR or prompt global strike. When considered under the jurisdiction of air law, aeroassisted maneuvers implemented in violation of a state-imposed no-fly zone would constitute a breach of international treaty.

Environmental Considerations

Occurring within the 50 – 90 km altitude regime, aeroassisted maneuvers place a TAV not only within potential foreign airspace, but also in the physical environment of the upper atmosphere. Of the various human space activities, space launch produces a high level of exhaust pollutants in the form of dust, the emission of toxic compounds such as aluminum oxide (from solid propellant), and the spraying of unburned liquid propellant like hydrazine. Although argued by many to have a negligible cumulative effect on atmospheric degradation, the burning of rocket propellant – whether solid or liquid in composition – in the upper atmosphere has been demonstrated to deteriorate the ozone layer and chemically contaminate the water cycle.¹⁷⁹ Not chemically destructive, the release of water as an exhaust by-product can interfere with ionospheric conditions, thus disrupting the transmission of wireless communications.¹⁸⁰

¹⁷⁸ Joseph T. Stanik, *El Dorado Canyon: Reagan’s Undeclared War with Qaddafi* (Annapolis, MD: Naval Institute Press, 2003), 145-146.

¹⁷⁹ Lotta Viikari, *The Environmental Element in Space Law: Assessing the Present and Charting the Future* (Leiden, The Netherlands: Koninklijke Brill NV, 2008), 29-31.

¹⁸⁰ *Ibid.*, 31.

While the aeroglide type of aeroassisted maneuver only performs thruster burns in space, the aerobang and aerocruise alternatives produce a steady thrust throughout the trans-atmospheric trajectory. Even though a TAV's propulsion system burns liquid rather than solid propellants and, therefore, produces fewer pollutants, exhaust by-products are continuously injected into the airspace when an aerobang or aerocruise maneuver is implemented. As a result, can a state deny the infringement of its airspace by an aeroassisted maneuver due to environmental considerations? If an aerobang or aerocruise maneuver is considered within the jurisdiction of air law, does the operator of the TAV assume sole liability for any environmental impact of the maneuver?

Summary and Conclusion

The continued engineering development of TAVs will undoubtedly require the air and space law challenges of aeroassisted maneuvers to be formally addressed due to ongoing debate associated with the prospect of airspace and outer space delimitation. Occurring within the 50 – 90 km altitude regime, spatialism dictates that TAVs implementing an aeroassisted maneuver are subject to air law. From the functionalist perspective, however, the legal delimitation of air and space law becomes ambiguous with arguments that can identify a TAV as either an aircraft or space object. For many states, to include the Lebanon, the Syrian Arab Republic, and Turkey, the stance is clear: A TAV traversing foreign airspace during an aeroassisted maneuver is subject to air law. For states like the Czech Republic though, ambiguity resurfaces with the view that air law only applies to “objects resembling [spaceplanes], but not to objects resembling Space Shuttles.”¹⁸¹

¹⁸¹ Gorove, 21-22.

Due to the unique hybrid characteristics of not only aeroassisted maneuvers, but also TAVs, one viable solution option is to spatially establish an exclusive zone of operation for TAVs between the maximum operating ceiling of conventional aircraft and the lowest achievable orbit for satellites. Defined as a *sui generis* zone, the approximate altitude regime of 21 – 96 km would permit the freedom of operation of TAVs (within peaceful bounds) and officially delimit the boundaries of both airspace sovereignty and outer space.¹⁸² A propitious compromise, such an exclusive zone for TAVs could forestall the onset of legal challenges for nascent commercial and national ventures seeking to implement aeroassisted maneuvers.

¹⁸² Su, 92.

VIII. Conclusions and Recommendations

Conclusions of Research

Once verified by duplicating the re-entry trajectory of the Apollo 10 command module capsule, the trajectory dynamics model was used to determine the terrestrial and LEO reachability potential of aeroassisted maneuvers, specifically skip entry and descent-boost. During the terrestrial reachability study, a series of skip entry and exo-atmospheric planar phasing and simple plane change maneuvers were first simulated to establish the time-of-arrival and ΔV required for each respective maneuver to overfly specific ground targets located at high-, medium-, and low-latitudes. For the sample target of Moscow, it was demonstrated that skip entry maneuvers provide the fastest time-of-arrival at a low ΔV when compared with the planar phasing and simple plane change maneuver alternatives. While the ΔV for the simple plane change is lower than most phasing maneuvers for targets such as Moscow and Gibraltar, the equatorial target of Pontianak, Indonesia illustrated that the choice of ground target can have a detrimental impact on ΔV with values approaching 8.0 km/s for a single simple plane change maneuver. For a limited yet diverse set of sample ground targets, skip entry maneuvers are shown to require a total ΔV less than 0.5 km/s and consistently provide responsive mission execution in terms of target time-of-arrival.

While the ground target over-flight simulations assumed a notional TAV design with a hypersonic lift-to-drag ratio of $L/D = 6$, the second phase of the terrestrial reachability study sought to determine aeroassisted maneuver performance by optimizing both TAV and maneuver trajectory design. Of the various optimization algorithms available, to include pseudospectral and meta-heuristic methods, the Design of Experiments method of orthogonal arrays was employed since it provides for an augmented exploration of the objective space with the ability to perform

main effects as well as Pareto front analysis. Initially, three multiple-objective optimization problems (MOPs) were devised from in order to obtain optimal designs for both a TAV and skip entry trajectory: (1) $\{\max(\Delta i), \min(\Delta V_{Total})\}$; (2) $\{\max(\Delta i), \max(h_{recirc})\}$; and (3) $\{\min(\Delta V), \max(h_{recirc})\}$. The first, or primary MOP sought maximize terrestrial reachability in the form of inclination change, Δi , while minimizing ΔV ; the secondary and tertiary MOPs focused on maximizing post-maneuver re-circularization altitude while maximizing reachability and minimizing ΔV , respectively. Although the maximization of re-circularization altitude permits the execution of subsequent maneuvers and, therefore, a greater terrestrial reachability potential due to higher orbital potential energy, Pareto analysis revealed that all three MOPs could not be satisfied without sacrificing either Δi , ΔV , or re-circularization altitude. Consequently, the optimization was restricted to the primary MOP and the combined main effects and Pareto front analysis yielded the following optimal TAV and skip entry trajectory design:

Table 8.1. Optimal TAV Design and Trajectory from DOE Analysis

Mass, kg	2000
Planform Area, m ²	18.5
Drag Coefficient	0.5
Lift Coefficient	3.0
Initial Altitude, km	1000
Perigee Altitude, km	86.75
Bank Angle	−90 deg

Starting from a circular reference orbit with an inclination of 37.84 deg, the optimal TAV design can achieve a maximum inclination change of $\Delta i = 19.91$ deg for $\Delta V = 0.345$ km/s if re-circularization occurs at the skip apogee altitude of 131.2 km. By performing a Hohmann transfer following the aeroassisted maneuver in order to re-circularize at a sample

higher altitude of 500 km, the total ΔV required to perform both maneuvers is 0.806 km/s. Without an orbit-raising transfer, the orthogonal array optimization demonstrated that a skip entry maneuver requires approximately 50-85% less ΔV than a simple plane change to achieve a maximum inclination change of $\Delta i = 19.91$ deg.

As an alternative to skip entry, the descent-boost type of aeroassisted maneuvers was used to perform the LEO reachability study. For a single TAV design with $L/D = 6$, an initial maneuver performance sensitivity study indicated the total descent-boost maneuver ΔV is a strong function of both initial flight-path angle and inclination, with ΔV increasing as the magnitude of these respective parameters increases. Utilizing MP number analysis, the combined Hohmann transfer was deemed a more effective maneuver option for injection into orbits such as Molniya, with the descent-boost maneuver generally requiring a greater ΔV expenditure due to the initial ΔV impulse performed to alter TAV trajectory flight-path angle and orbital velocity. Although requiring a longer time-of-flight for orbit injection than bi-elliptic transfers, descent-boost maneuvers commencing from the initial altitudes of 1000, 1100, and 1200 km are shown here to require 6-12% less ΔV for injection into circular orbits with altitudes less than 650 km.

Through the pursuance of both trajectory- and optimization-centric performance analysis, aeroassisted maneuvers in the form of skip entry and descent-boost have been demonstrated in several cases to require a lower ΔV expenditure than exo-atmospheric maneuvers in order to achieve terrestrial and LEO reachability. Despite potential air and space law challenges arising due to current ambiguities surrounding atmospheric delimitation and the question of airspace sovereignty, aeroassisted maneuvers provide several implicit capabilities not readily available for conventional exo-atmospheric maneuvers. For skip entry, maneuver unpredictability is conceivable by penetrating the upper atmosphere to utilize aerodynamic forces to change orbital

states such as inclination and semi-major axis. With descent-boost maneuvers, the TAV cannot only perform an orbital inspection upon transiting skip apogee during orbit injection, but also provide a level of unpredictability since the descent-boost trans-atmospheric trajectories feature a decaying elliptical flight-path which produces multiple orbit injection possibilities.

Significance of Research

Aeroassisted maneuvers – specifically skip entry and descent-boost – provide a viable alternative to exo-atmospheric maneuvers for the alteration of orbital states and the completion of user-defined mission objectives linked to ground target over-flight as well as LEO injection. Besides trajectory-centric design analysis, the Design of Experiments method of orthogonal arrays has been demonstrated as an advantageous means of optimizing both TAV and skip entry maneuver trajectory for a multi-objective optimization problem (MOP) through the ability to perform main effects and Pareto front analysis. Utilized in the LEO reachability study, the concept of the Maneuver Performance (MP) number was introduced as a dimensionless means of comparative effectiveness analysis for exo- and trans-atmospheric maneuvers. Based on inherent analysis limitations of a single formulation, two versions of the effectiveness ratio are provided; the first is applicable to maneuvers between non-equal initial and final orbital altitudes, while the second accounts for phasing maneuvers in which the initial and final orbital altitudes are equal. Finally, all aeroassisted maneuver simulations comprising the present research used a piecewise-continuous atmospheric density function that was developed to model the MSIS-E-90 density profile by incorporating three separate altitude-delimited models: (1) Exponential density for $h \in [0, 84]$ km; (2) scale height-varying density for $h \in [84, 120]$ km; and (3) power regression density for $h \in [120, 1000]$ km.

Recommendations for Future Research

Although providing an assessment of aeroassisted maneuver performance in terms of terrestrial and LEO reachability, the present research features limitations based on the simulation simplifying assumptions and the circumscribed investigation of only the skip entry and descent-boost types of aeroassisted maneuvers. As a result, recommendations for future research are enumerated below:

1. Investigate the effect of induced drag on aeroassisted maneuver performance by utilizing a vehicle-specific drag polar rather than constant aerodynamic coefficients.
2. Conduct a comparative analysis of aeroassisted skip entry and exo-atmospheric maneuvers for ground target over-flight with variable initial RAAN.
3. For the Design of Experiments optimization segment of research, expand the variable bank angle analysis to account for greater inverted-TAV motion with $\sigma \in [-160, 0]$ deg so as to increase duration of trans-atmospheric flight and maximize inclination change.
4. Explore the optimal implementation of aerobang and aerocruise as alternatives to skip entry and descent-boost. Specifically, investigate if an optimal location exists along the trans-atmospheric trajectory at which to commence continuous thrusting in order to maximize inclination change while minimizing ΔV .
5. Investigate the performance effects of conducting periodic impulsive thrusting along the trans-atmospheric trajectory instead of continuous thrusting for aerobang and aerocruise aeroassisted maneuvers.
6. Conduct a comparative analysis of aeroassisted maneuvers and Lambert transfers in terms of ΔV and time-of-flight performance. See Appendix D for a Lambert transfer solution algorithm.

Appendix A: Exo-Atmospheric Maneuver Algorithms

Initially described in Chapter I, the algorithms underpinning the Hohmann, combined Hohmann, bi-elliptic, and planar phasing exo-atmospheric maneuvers are given by the following:

Hohmann Transfer¹⁸³

Based on the assumption that both the parking (A) and mission (B) orbits are circular, the semi-major axis of the Hohmann transfer ellipse (T) is expressed by Eq. (A.1) in terms of the geocentric orbital radii r_A and r_B :

$$a_T = \frac{1}{2}(r_A + r_B) \quad (\text{A.1})$$

The tangential impulse ΔV_A required to inject the spacecraft into the perigee of the transfer ellipse from parking orbit (A) is defined by the circular orbit velocity of (A) and the velocity of the elliptical trajectory corresponding with the impulse location:

$$V_A = \sqrt{\frac{\mu}{r_A}} \quad V_{T,A} = \sqrt{\mu \left(\frac{2}{r_A} - \frac{1}{a_T} \right)} \quad (\text{A.2})$$

$$\Delta V_A = |V_A - V_{T,A}| \quad (\text{A.3})$$

Similarly, the tangential impulse ΔV_B required to re-circularize the spacecraft at the intersection of the transfer ellipse apoapsis and the mission orbit (B) is given by:

$$V_B = \sqrt{\frac{\mu}{r_B}} \quad V_{T,B} = \sqrt{\mu \left(\frac{2}{r_B} - \frac{1}{a_T} \right)} \quad (\text{A.4})$$

$$\Delta V_B = |V_B - V_{T,B}| \quad (\text{A.5})$$

The summation of the two impulse burns yields the total ΔV for the Hohmann transfer:

$$\Delta V_{Total} = \Delta V_A + \Delta V_B = |V_A - V_{T,A}| + |V_B - V_{T,B}| \quad (\text{A.6})$$

¹⁸³ Vallado, 327.

Finally, the time-of-flight of the Hohmann transfer is defined as one half of the period of the transfer ellipse:

$$t_T = \pi \sqrt{\frac{a_T^3}{\mu}} \quad (\text{A.7})$$

As an alternative, Vladimir A. Chobotov in *Orbital Mechanics* defines the time-of-flight as a function of the parking and mission orbital radii:¹⁸⁴

$$t_T = \pi \sqrt{\frac{r_A^3}{\mu}} \cdot \left\{ \frac{1}{2^{5/2}} \left(1 + \frac{r_B}{r_A} \right)^{3/2} \right\} \quad (\text{A.8})$$

Combined Hohmann Transfer¹⁸⁵

For cases in which the parking (A) and mission (B) orbits are circular and non-coplanar, the combined Hohmann transfer is utilized to change both inclination and semi-major axis. Initially, the transfer ellipse semi-major axis is given by Eq. (A.1) and the velocities associated with the parking and mission orbits as well as the transfer ellipse perigee and apogee locations are given by Eqs. (A.2) and (A.4), respectively. In order to minimize the total ΔV for the maneuver, the inclination change is incorporated into the impulse burns at the transfer ellipse periapsis and apoapsis. At the transfer ellipse injection point in parking orbit (A), the amount of inclination change is expressed by:

$$\Delta i_A = s \Delta i \quad (\text{A.9})$$

Likewise, the amount of inclination change to perform during re-circularization at mission orbit (B) is:

$$\Delta i_B = (1 - s) \Delta i \quad (\text{A.10})$$

¹⁸⁴ Chobotov, 95.

¹⁸⁵ Vallado, 354-355.

One option of determining the “best” amount of inclination change to perform at each transfer burn consists of iterating the transcendental equation of $\sin(s\Delta i)$ given by Eq. (A.11):

$$\sin(\Delta i_A) = \frac{\Delta V_A V_B V_{T,B} \sin(\Delta i_B)}{\Delta V_B V_A V_{T,A}} \quad (\text{A.11})$$

A second option, which is used for descent-boost maneuver comparative analysis in Chapter VII, involves the following non-iterative analytic approximation:

$$s \approx \frac{1}{\Delta i} \tan^{-1} \left[\frac{\sin(\Delta i)}{R^{3/2} + \cos(\Delta i)} \right] \quad (\text{A.12})$$

where $R = r_B/r_A$. Utilizing the *Law of Cosines*, the transfer burn impulses ΔV_A and ΔV_B are given by Eqs. (A.13) and (A.14):

$$\Delta V_A = \sqrt{V_A^2 + V_{T,A}^2 - 2V_A V_{T,A} \cos(\Delta i_A)} \quad (\text{A.13})$$

$$\Delta V_B = \sqrt{V_B^2 + V_{T,B}^2 - 2V_B V_{T,B} \cos(\Delta i_B)} \quad (\text{A.14})$$

The summation of the two transfer burns yields the total ΔV for the combined Hohmann transfer:

$$\Delta V_{Total} = \Delta V_A + \Delta V_B \quad (\text{A.15})$$

Similar to the Hohmann transfer, the time-of-flight of the combined Hohmann transfer is expressed by Eq. (A.7).

Bi-Elliptic Transfer¹⁸⁶

Unlike the preceding maneuvers, the design of the bi-elliptic transfer features two transfer ellipses. The first ellipse extends from the parking orbit (A) to an intermediate orbit (B), which is greater in altitude than the mission orbit (C). The semi-major axis of the first ellipse is given by:

¹⁸⁶ Ibid., 328.

$$a_1 = \frac{1}{2}(r_A + r_B) \quad (\text{A.16})$$

With a semi-major axis defined by Eq. (A.17), the second ellipse extends from the intermediate orbit (B) to the mission orbit (C):

$$a_2 = \frac{1}{2}(r_B + r_C) \quad (\text{A.17})$$

The velocities associated with the bi-elliptic transfer are expressed as:

$$V_A = \sqrt{\frac{\mu}{r_A}} \quad V_{1,A} = \sqrt{\mu \left(\frac{2}{r_A} - \frac{1}{a_1} \right)} \quad (\text{A.18})$$

$$V_{1,B} = \sqrt{\mu \left(\frac{2}{r_B} - \frac{1}{a_1} \right)} \quad V_{2,B} = \sqrt{\mu \left(\frac{2}{r_B} - \frac{1}{a_2} \right)} \quad (\text{A.19})$$

$$V_C = \sqrt{\frac{\mu}{r_C}} \quad V_{2,C} = \sqrt{\mu \left(\frac{2}{r_C} - \frac{1}{a_2} \right)} \quad (\text{A.20})$$

where V_A is the velocity of parking orbit (A), $V_{1,A}$ is the perigee velocity of the first transfer ellipse at (A), $V_{1,B}$ is the apogee velocity of the first transfer ellipse at the intermediate orbit (B), $V_{2,B}$ is the apogee velocity of the second transfer ellipse at (B), $V_{2,C}$ is the perigee velocity of the second transfer ellipse at the mission orbit (C), and V_C is the velocity of (C).

The total ΔV and the time-of-flight are given by Eqs. (A.21) and (A.22), respectively:

$$\Delta V_{Total} = \Delta V_A + \Delta V_B + \Delta V_C = |V_1 - V_{1,A}| + |V_{1,B} - V_{2,B}| + |V_C - V_{2,C}| \quad (\text{A.21})$$

$$t_{Total} = \pi \sqrt{\frac{a_1^3}{\mu}} + \pi \sqrt{\frac{a_2^3}{\mu}} \quad (\text{A.22})$$

Phasing Maneuvers

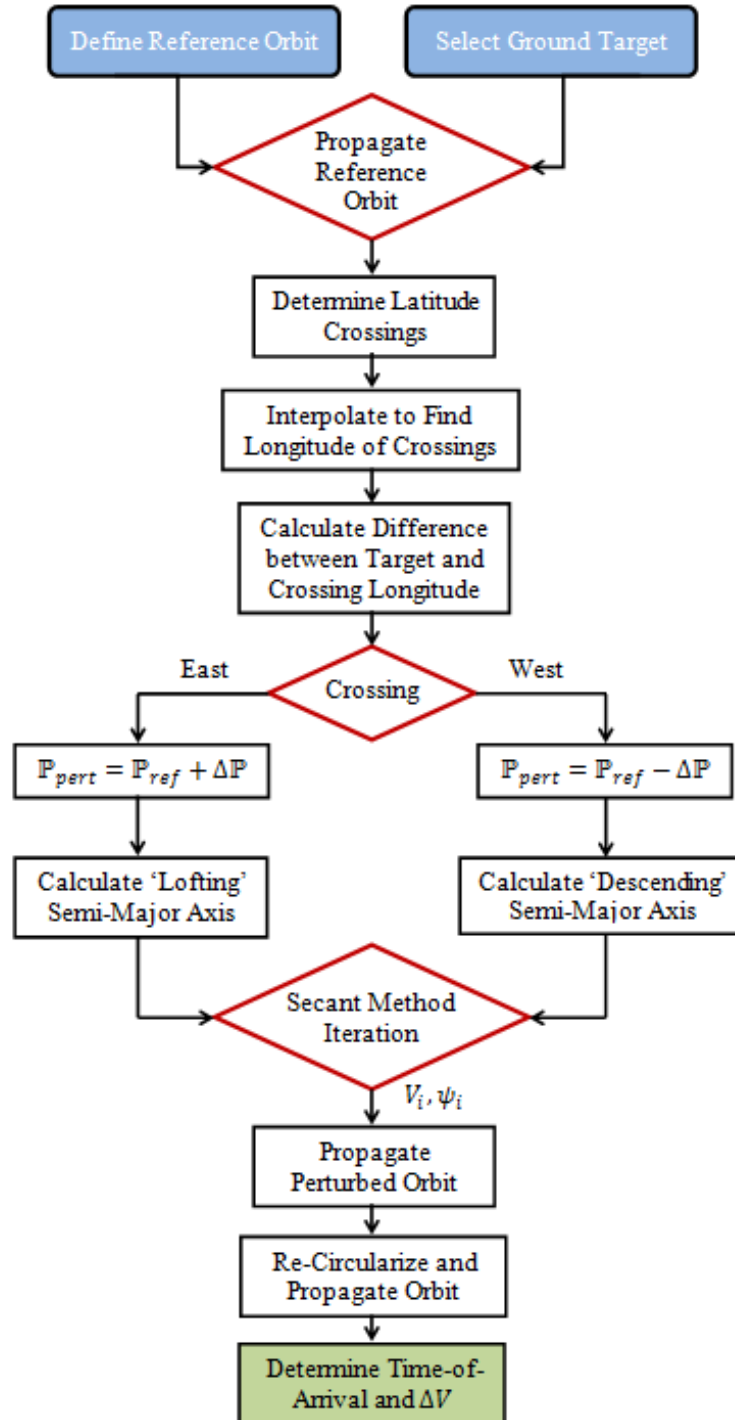


Figure A.1. Phasing Maneuver Flowchart

Appendix B: Geodesic Equation Formulation

In his paper “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Applications of Nested Equations,” Thaddeus Vincenty presented what he termed as “compact formulae” for both the direct and inverse solutions of geodesics. For the purposes of the present research, only the direct solution algorithm are presented.¹⁸⁷ Defined in terms of an ellipsoidal planetary model, the direct solution is a function of longitude and geodetic latitude of each location point, as well as the major (a) and minor (b) semi-axes of the ellipsoid. With these inputs, the reduced latitude, U , is calculated in terms of geodetic latitude and the flattening parameter: $f = (a - b)/a$.

$$\begin{aligned} U_1 &= \tan^{-1}((1 - f) \tan \phi_1) \\ U_2 &= \tan^{-1}((1 - f) \tan \phi_2) \end{aligned} \tag{B.1}$$

The term “reduced” indicates that the latitude U is measured on an auxiliary sphere centered and located coincident with an ellipsoidal model. From the longitudinal components of each location coordinate set, the quantity λ is determined from measurements on the auxiliary sphere:

$$\lambda = \theta_2 - \theta_1 \tag{B.2}$$

By initially setting $\lambda = L$, where L is the difference in longitude on the ellipsoid, Eqs. (B.3) – (B.10) are solved iteratively until λ converges to a specified error tolerance, e.g. 1.0×10^{-12} :

$$\sin \sigma = \sqrt{(\cos U_2 \sin \lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos \lambda)^2} \tag{B.3}$$

$$\cos \sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos \lambda \tag{B.4}$$

$$\sigma = \tan^{-1} \left(\frac{\sin \sigma}{\cos \sigma} \right) \tag{B.5}$$

¹⁸⁷ Thaddeus Vincenty, “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Applications of Nested Equations,” *Survey Review* XXII 176 (1975): 88-90.

$$\sin \alpha = \frac{\cos U_1 \cos U_2 \sin \lambda}{\sin \sigma} \quad (\text{B.6})$$

$$\cos^2 \alpha = 1 - \sin^2 \alpha \quad (\text{B.7})$$

$$\cos 2\sigma_m = \cos \sigma - \frac{2 \sin U_1 \sin U_2}{\cos^2 \alpha} \quad (\text{B.8})$$

$$C = \frac{f}{16} \cos^2 \alpha \cdot [4 + f(4 - 3 \cos^2 \alpha)] \quad (\text{B.9})$$

$$\lambda = L + (1 - C)f \sin \sigma \cdot \{\sigma + C \sin \sigma [\cos 2\sigma_m + C \cos \sigma (-1 + 2 \cos^2 2\sigma_m)]\} \quad (\text{B.10})$$

From the equations above, α is the azimuth of the geodesic, σ is the angular distance between the coordinate locations on the auxiliary sphere, and σ_m is the angular distance from the equator to the midpoint of the geodesic on the auxiliary sphere.

With convergence attained, the following equations are solved in succession until the geodesic distance s is calculated in Eq. (B.15):

$$u^2 = \cos^2 \alpha \cdot \left(\frac{a^2 - b^2}{b^2} \right) \quad (\text{B.11})$$

$$A = 1 + \frac{u^2}{16384} \{4096 + u^2[-768 + u^2(320 - 175u^2)]\} \quad (\text{B.12})$$

$$B = \frac{u^2}{1024} \{256 + u^2[-128 + u^2(74 - 47u^2)]\} \quad (\text{B.13})$$

$$\begin{aligned} \Delta\sigma = B \sin \sigma \cdot & \left\{ \cos 2\sigma_m + \frac{B}{4} \right. \\ & \cdot \left[\cos \sigma (-1 + 2 \cos^2 2\sigma_m) - \frac{B}{6} \right. \\ & \cdot \left. \cos 2\sigma_m (-3 + 4 \sin^2 \alpha)(-3 + 4 \cos^2 2\sigma_m) \right] \left. \right\} \end{aligned} \quad (\text{B.14})$$

$$s = bA(\sigma - \Delta\sigma) \quad (\text{B.15})$$

For all equations, A , B , and C represent intermediate variables.

Appendix C: TLE Guide

Updated twice daily, the North American Aerospace Defense Command (NORAD) provides an inventory of orbiting objects in the form of Two-Line Element (TLE) sets for each space object. The TLEs contain information pertaining to the position of the object within its orbit as well as the orbit position relative to the Earth-Centered Inertial (ECI) reference frame.¹⁸⁸ In his book *Satellites: Orbits and Missions*, Michel Capderou presents a general TLE format and element description as adapted in Table C.1. Note that the letter *A* in the general TLE refers to “alphabetical character,” while *N* is a “numerical character.”

Table C.1 General TLE and Element Description¹⁸⁹

<pre> AAAAAAAAAAAAAAAAAAAAAAAAA 1 NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNNN-N +NNNNN-N N NNNNN 2 NNNNN NNN.NNNN NNN.NNNN NNNNNNNN NNN.NNNN NNN.NNNN NN.NNNNNNNNNNNNNNN </pre>		
<i>Line</i>	<i>Column</i>	<i>Description</i>
1	01	Line number
1	03-08	Satellite number with classification
1	10-17	International designator
1	19-32	Epoch year; day of year; fraction of day
1	34-43	First time derivative of mean motion, n
1	45-52	Second time derivative of n (decimal point assumed)
1	54-61	Drag term (decimal point assumed)
1	63-69	Ephemeris type; element number; checksum (modulo 10)
2	01-07	Line number; satellite number without classification
2	09-16	Inclination, i (degrees)
2	18-25	Right ascension of the ascending node, Ω (degrees)
2	27-33	Eccentricity, e (decimal point assumed)
2	35-42	Argument of perigee, ω (degrees)
2	44-51	Mean anomaly, M (degrees)
2	53-63	Mean motion, n (revolutions per day)
2	64-69	Revolution number at epoch; checksum (modulo 10)

¹⁸⁸ Michel Capderou, *Satellites: Orbits and Missions* (Paris, France: Springer-Verlag France, 2005), 254.

¹⁸⁹ Ibid., 254-255.

An example TLE for the Molniya 3-42 communications satellite is given in Figure C.1.

While four of the six standard Keplerian elements are provided in the TLE (e , i , Ω , ω), the remaining elements of semi-major axis, a , and true anomaly, v , must be calculated utilizing the mean motion and mean anomaly data.

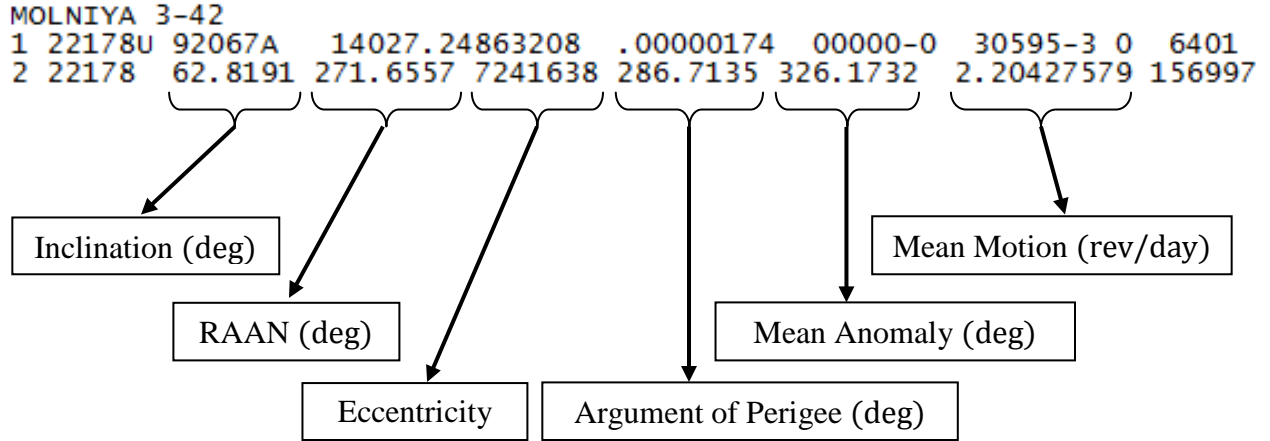


Figure C.1. Element Mapping for Molniya 3-42 Example TLE

As a function of mean motion, the semi-major axis is given by the following:¹⁹⁰

$$a = \left(\frac{\mu}{n^2} \right)^{1/3} \quad (\text{C.1})$$

The true anomaly as shown in Eq. (C.3) is calculated by first solving Kepler's equation in Eq.

(C.2) via a Newton-Rhapson iteration for the eccentric anomaly, E :¹⁹¹

$$M = E - e \sin E \quad (\text{C.2})$$

$$v = \cos^{-1} \left(\frac{\cos E - e}{1 - e \cos E} \right) \quad (\text{C.3})$$

¹⁹⁰ Vallado, 31.

¹⁹¹ Ibid., 54-55, 73.

Appendix D: Lambert Algorithm

Originally formulated in 1761, Swiss mathematician Johann H. Lambert's eponymous problem seeks to determine the orbit between two known position vectors and represents a two-point boundary value problem.¹⁹² Overall, Lambert's problem permits several different situations in astrodynamics to be examined, from initial orbit determination based on a preliminary set of observation vectors (e.g. Gauss's efforts to determine the orbit of the planetoid Ceres), to intercept and rendezvous between position vectors either within the same orbit, or in two separate orbits.¹⁹³ Regardless of the application, numerous solution algorithms exist for Lambert's problem. In his book *Fundamentals of Astrodynamics and Applications*, Vallado not only presents Gauss's solution,¹⁹⁴ but also the power series solution developed by Thorne,¹⁹⁵ a method utilizing universal variables,¹⁹⁶ and an overview of Battin's method.¹⁹⁷ The complete derivation of the last method can be found in Richard H. Battin and Robin M. Vaughn's original paper "An Elegant Lambert Algorithm."¹⁹⁸ Besides Battin and the other methods discussed by Vallado, other examples of Lambert algorithms include Gim J. Der's formulation of a multi-revolution analytic solution and R. H. Gooding's approach based on Halley's cubic iteration method.¹⁹⁹

¹⁹² Vallado, 420; Michael O'Leary, *Revolutions of Geometry* (Hoboken, NJ: John Wiley & Sons, Inc., 2010), 353; Gim J. Der, "The Superior Lambert Algorithm" (paper presented at the *Advanced Maui Optical and Space Surveillance Technologies Conference*, Wailea, Maui, HI, 13-16 September 2011): 4.

¹⁹³ Vallado, 472, 495.

¹⁹⁴ Ibid., 472-476.

¹⁹⁵ Ibid., 476-485.

¹⁹⁶ Ibid., 485-490.

¹⁹⁷ Ibid., 490-494.

¹⁹⁸ Richard H. Battin and Robin M. Vaughn, "An Elegant Lambert Algorithm," *Journal of Spacecraft and Rockets* 7, no. 6 (1984): 662-670.

¹⁹⁹ Der, 1-28; R. H. Gooding, "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem," *Celestial Mechanics and Dynamical Astronomy* 48, no. 2 (1990): 145-165.

For the present research, a variation of the universal variable algorithm utilizing Newton's method is outlined as described in Tewari's text *Atmospheric and Spaceflight Dynamics*.²⁰⁰ In Tewari's algorithm, the initial and final velocities vectors describing the transfer orbit are determined by first calculating the transfer angle, ϕ , between the two position vectors:

$$\phi = \begin{cases} \cos^{-1} \left(\frac{\vec{r}_1 \cdot \vec{r}_2}{r_1 r_2} \right) & , \alpha > 0 \\ 2\pi - \cos^{-1} \left(\frac{\vec{r}_1 \cdot \vec{r}_2}{r_1 r_2} \right) & , \alpha < 0 \end{cases} \quad (\text{D.1})$$

where $\alpha = \hat{k} \cdot (\vec{r}_1 \times \vec{r}_2)$ and $\hat{k} = [0 \ 0 \ 1]^T$. With the transfer angle defined in the appropriate quadrant, the variable A is calculated by:

$$A = \sin \phi \sqrt{\frac{r_1 r_2}{1 - \cos \phi}} \quad (\text{D.2})$$

Initial values for the auxiliary variables x, y, z are determined by first assuming a value for z , which Tewari defines as "usually a small, positive number," such as $z = 0.01$. Equations for x and y are:

$$y = r_1 + r_2 - \frac{A}{\sqrt{C(z)}} (1 - zS(z)) \quad (\text{D.3})$$

$$x = \sqrt{\frac{y}{C(z)}} \quad (\text{D.4})$$

The variables $C(z)$ and $S(z)$ represent *Stumpff functions*, which are expressed by the two infinite series in Eq. (D.5); for a discussion of the mathematical properties associated with Stumpff functions, see Bate, Mueller, and White's *Fundamentals of Astrodynamics*.²⁰¹

²⁰⁰ Tewari, 144-147.

²⁰¹ Roger R. Bate, Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics* (New York, NY: Dover Publications, Inc., 1971), 196.

$$\begin{aligned}
C(z) &= \frac{1}{2!} - \frac{z}{4!} + \frac{z^2}{6!} - \dots = \sum_{i=0}^{\infty} \frac{(-z)^i}{(2i+2)!} \\
S(z) &= \frac{1}{3!} - \frac{z}{5!} + \frac{z^2}{7!} - \dots = \sum_{i=0}^{\infty} \frac{(-z)^i}{(2i+3)!}
\end{aligned} \tag{D.5}$$

The derivation of the Eq. (D.3) is given as a coda to this Appendix.

Based on the preceding values for x and z , an initial value for the transfer time, t , is then calculated with the following equation:

$$t = \frac{1}{\sqrt{\mu}} (S(z) \cdot x^3 + A\sqrt{C(z)} \cdot x) \tag{D.6}$$

Once determined, the initial values for x, y, z, t serve to initiate a Newton's method algorithm based on the following cubic equation, to which Eq. (D.4) is a solution:

$$\sqrt{\mu}(t_f - t_i) = A\sqrt{C(z)}x + S(z)x^3 \tag{D.7}$$

Substituting Eq. (D.7) into the Newton sequence yields:²⁰²

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)} = x_n - \frac{S(z_n) \cdot (x_n)^3 + A\sqrt{C(z_n)} \cdot (x_n) - \sqrt{\mu}(t_n - t_i)}{3 \cdot S(z_n) \cdot (x_n)^2 + A\sqrt{C(z_n)}} \tag{D.8}$$

Iterative values for y, z, t are given by:

$$\begin{aligned}
y_{n+1} &= C(z_n) \cdot (x_{n+1})^2 \\
z_{n+1} &= \frac{1}{S(z_n)} \left(1 - \frac{C(z_n)}{A} \right) (r_1 + r_2 - y_{n+1}) \\
t_{n+1} &= \frac{1}{\sqrt{\mu}} \left(S(z_n) \cdot (x_{n+1})^3 + A\sqrt{C(z_n)} \cdot (x_{n+1}) \right)
\end{aligned} \tag{D.9}$$

Updated values for the Stumpff functions $C(z_{n+1}), S(z_{n+1})$ are calculated utilizing Eq. (D.5).

²⁰² C. T. Kelley, *Solving Nonlinear Equations with Newton's Method* (Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2003), 2; Tewari, 147.

Once the solution for z has converged within a specified tolerance, then the final value for y is used to solve for the Lagrange coefficients:

$$\begin{aligned} f &= 1 - \frac{y}{r_1} \\ g &= A \sqrt{\frac{y}{\mu}} \\ \dot{g} &= 1 - \frac{y}{r_2} \\ \dot{f} &= \frac{1}{g} (f \dot{g} - 1) \end{aligned} \tag{D.10}$$

Finally, the Lagrange coefficients enable the determination of the initial and final velocity vectors of the transfer orbit, \vec{v}_1, \vec{v}_2 :

$$\vec{v}_1 = \frac{1}{g} (\vec{r}_2 - f \vec{r}_1) \tag{D.11}$$

$$\vec{v}_2 = \dot{f} \vec{r}_1 + \dot{g} \vec{v}_1 \tag{D.12}$$

Alternatively, Eq. (D.12) can be written as:

$$\vec{v}_2 = \frac{1}{g} (\dot{g} \vec{r}_2 - \dot{f} \vec{r}_1) \tag{D.13}$$

Derivation of Equation for Auxiliary Variable y

The equation for y , or Eq. (D.3), is derived by first substituting the expressions for the f, g, \dot{g} Lagrange coefficients in Eq. (D.10) into the relationship $(f \dot{g} - g \dot{f} = 1)$ to yield:

$$\left(1 - \frac{y}{r_1}\right) \left(1 - \frac{y}{r_2}\right) - \left(A \sqrt{\frac{y}{\mu}}\right) \dot{f} = 1$$

Solving for the fourth Lagrange coefficient, \dot{f} , gives:

$$\dot{f} = \frac{\left(1 - \frac{y}{r_1}\right)\left(1 - \frac{y}{r_2}\right) - 1}{A\sqrt{\frac{y}{\mu}}} = \frac{\left(\frac{y^2}{r_1 r_2} - \frac{y}{r_1} - \frac{y}{r_2}\right)}{A\sqrt{\frac{y}{\mu}}} = \frac{y}{A\sqrt{\frac{y}{\mu}}} \left(\frac{y - r_2 - r_1}{r_1 r_2}\right) \quad (\text{D.14})$$

In terms of the variable x and the Stumpff function $S(z)$, an alternate formulation of Eq. (D.14) is given by:²⁰³

$$\dot{f} = \frac{\sqrt{\mu}}{r_1 r_2} (xzS(z) - x) \quad (\text{D.15})$$

where $z = x^2/a$, and a is the semi-major axis of the transfer orbit. By setting Eq. (D.14) equal to Eq. (D.15), an equation for y can be determined via the following algebra:

$$\begin{aligned} \frac{y}{A\sqrt{\frac{y}{\mu}}} \left(\frac{y - r_2 - r_1}{r_1 r_2}\right) &= \frac{\sqrt{\mu}}{r_1 r_2} (xzS(z) - x) \\ \frac{y\sqrt{\mu}}{A\sqrt{y}} \left(\frac{1}{r_1 r_2}\right) (y - r_2 - r_1) &= \frac{x\sqrt{\mu}}{r_1 r_2} (zS(z) - 1) \\ \frac{\sqrt{y}}{A} (y - r_2 - r_1) &= x(zS(z) - 1) \end{aligned}$$

If $x = \sqrt{y/C(z)}$, then $\sqrt{y} = x\sqrt{C(z)}$:

$$\begin{aligned} \frac{x\sqrt{C(z)}}{A} (y - r_2 - r_1) &= x(zS(z) - 1) \\ y - r_2 - r_1 &= \frac{A}{\sqrt{C(z)}} (zS(z) - 1) \\ y &= r_1 + r_2 - \frac{A}{\sqrt{C(z)}} (1 - zS(z)) \end{aligned} \quad (\text{D.16})$$

²⁰³ Tewari, 144.

Appendix E: MATLAB® Code for Trajectory Dynamics Model

Table E.1. m-File Classification for Trajectory Dynamics Model

Filename	File Type	Description
Maneuver_MainFunction	Function	Core module
EventFunction	Function	Solver stopping condition
Maneuver_SubFile	Function	Supports operation of solver
AtmosModel	Function	Atmospheric model
AtmosModel_PostAnalysis	Function	Atmos. model for post-processing
EntryEOM_Complete	Function	Equations of motion with T, ω_{\oplus}
EntryEOM_Simple	Function	Equations of motion without T, ω_{\oplus}
EntryEOM_Euler	Function	Equations of motion without T, ω_{\oplus}
GravityModel	Function	Gravity model
VehicleSpecs	Function	Spacecraft model
WGS84Constants	Function	Planetary constants

The core module of the Trajectory Dynamics Model contains the following options:

- *Spacecraft*: (1) TAV, (2) Apollo 10 capsule, or (3) various notional satellite designs.
- *Equations of Motion*: (1) “Complete” six-state set which includes thruster modeling and planetary rotation, or (2) “simple” six-state set which assumes a non-thrusting vehicle and non-rotating planetary model.
- *Planetary Rotation*: (1) Activated, or (2) de-activated.
- *Bank Angle Control Input*: (1) Constant bank angle throughout trajectory, or (2) time-dependent bank angle profile.
- *Differential Equation Solver*: (1) MATLAB® “ODE45” solver, or (2) Euler integration.

The former option supports both the spherical and J_2 -gravity models, while the latter supports only the spherical gravity model.

Maneuver_MainFunction.m

```
function [t,traj_param] = Maneuver_MainFunction(Choice_1,Choice_2,...
        Choice_3,Choice_4,Choice_5,Choice_6,Time_Max,...
        r,V,lon,lat,fpa,heading,bank_angle)

global Vehicle_Choice EOM_Choice Gravity_Choice Omega_Choice
global BankAngle_Choice Solver_Choice bank
WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% User-Defined Input Definitions
Vehicle_Choice = Choice_1;
%1 = Notional Trans-Atmospheric Vehicle (m = 5000      kg, Cd = 0.5)
%2 = ESPA SPL Notional Satellite          (m =  200      kg, Cd = 2.2)
%3 = Primary Payload Notional Satellite (m = 1000      kg, Cd = 2.2)
%4 = Apollo 10 Command Module Capsule    (m = 5498.22 kg, Cd = 1.2569)
%5 = Apollo 10 CM Capsule w/ Alt. Cl/Cd (m = 5498.22 kg, Cd = 1.255)
%6 = Notional Satellite                  (m = 2000      kg, Cd = 3.0)

EOM_Choice = Choice_2;
%1 = "Complete" entry EOM (6 states, includes thrusting & rotation)
%2 = "Simple"   entry EOM (6 states, assumes non-thrusting & non-rotation)

Gravity_Choice = Choice_3;
%1 = Force Equations with spherical (Newtonian) gravity model
%2 = Force Equations with J2 gravity model

Omega_Choice = Choice_4;
%1 = Planetary rotation "activated"
%2 = Planetary rotation "de-activated"

BankAngle_Choice = Choice_5;
%1 = Constant bank angle throughout trajectory
%2 = Specified bank angle profile

Solver_Choice = Choice_6;
%1 = MATLAB ODE 45 with 6-state EOM
%2 = Euler Integration (only spherical gravity, 6-state EOM)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Bank Angle Conversion
bank = deg2rad(bank_angle);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Equation of Motion (EOM) Solver
if Solver_Choice == 1
    % MATLAB ODE Solver with 6-State EOM
    % options = odeset('Events',@EventFunction,'RelTol',1e-7);
    options = odeset('RelTol',1e-3); %,'MaxStep',1.0);
    traj_int = [r V lon lat fpa heading];
    [t,traj_param] = ode45(@Maneuver_SubFile,[0,Time_Max],traj_int,options);
```

```

elseif Solver_Choice == 2
    %% Euler Integration with 6-State EOM
    deltaT      = 1; %Simulation propagation time-step (s)
    [t,traj_param] = EntryEOM_Euler(r,V,lon,lat,fpa,heading,bank,...
                                   deltaT,Time_Max);
end

```

EventFunction.m

```

function [value,isterminal,direction] = EventFunction(t,traj_param)

global RE

value = traj_param(1) - RE;
isterminal = 1;
direction = -1;

```

Maneuver_SubFile.m

```

function Y = Maneuver_SubFile(t,traj_param)

global OmegaE Vehicle_Choice EOM_Choice Omega_Choice bank
global mass S_m2 Cd Cl

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Trajectory States
r = traj_param(1); %Radial position (km)
V = traj_param(2); %Velocity (km/s)

%Initial latitude (lat) and longitude (lon)
lon = traj_param(3); lat      = traj_param(4);
%Initial flight-path (fpa) and heading (psi) angles
fpa = traj_param(5); heading = traj_param(6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planetary Rotation Rate
if      Omega_Choice == 1
    OmegaRot = OmegaE; %Planetary rotation "activated"
elseif Omega_Choice == 2
    OmegaRot = 0;      %Planetary rotation "de-activated"
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
[Vehicle] = VehicleSpecs(Vehicle_Choice);

mass      = Vehicle.mass; %Mass (kg)
S_m2      = Vehicle.S_m2; %Planform area (m^2)
S          = S_m2/(1000^2); %Planform area (km^2)

```

```

Cd      = Vehicle.Cd;      %Drag coefficient
Cl      = Vehicle.Cl;      %Lift coefficient
Thrust  = 0;               %Thrust (kg.km/s^2)
epsT    = 0;               %Thrust vector angle (rad)
zetaT    = 0;               %Thrust vector angle (rad)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Equations of Motion (EOM)
if      EOM_Choice == 1
    Y = EntryEOM_Complete(r,V,lon,lat,fpa,heading,bank,...
                          OmegaRot,mass,S,Cd,Cl,Thrust,epsT,zetaT);
elseif EOM_Choice == 2
    Y = EntryEOM_Simple(r,V,lon,lat,fpa,heading,bank,mass,S,Cd,Cl);
end

```

AtmosModel.m

```

function [Rho] = AtmosModel(h_gd,AtmosModel_Choice)

global RE BetaH Rho0

WGS84Constants; %Loads global constants from external m-file

%Note: AtmosModel_Choice
%1 = Exponential density model
%2 = Combined density model (approximation of MSIS model)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Exponential Density Model (kg/km^3)
if      AtmosModel_Choice == 1
    Rho = Rho0.*exp(-BetaH.*h_gd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Combined Density Model      (kg/km^3)
%Note: Exponential Model:          h < 84 km
%      Scale Height (v1) Variation Model: 84 <= h <= 120 km
%      Power Model:                121 <= h <= 1000 km

elseif AtmosModel_Choice == 2
%Reference altitude (km)
h_i      = [67; 85; 99; 110];

%Reference density (kg/km^3)
Rho_i    = [1.4975e-4; 7.726e-6; 4.504e-7; 5.930e-8] * (1000)^3;

%Reference scale height (km)
Hi       = [6.6597; 4.979; 5.905; 8.731];

%Reference molecular scale temperature (K)
Tmi      = [222.8; 165.7; 195.6; 288.2];

```

```

%Atmospheric constant (K/km)
Constant_A = [0.1296385; 0.1545455; 0.1189286; 0.5925240];

%Atmospheric constant (K/km)
Constant_B = [4.044231; 0.0; 3.878571; 19.17964];

%Dimensionless parameters
deltaH      = (Constant_A.*RE)./Hi;
deltaTM     = (Constant_B.*RE)./Tmi;

%Altitude Sections
if h_gd <= 84 %Section 1: Exponential model
    Rho = Rho0.*exp(-BetaH.*h_gd);

elseif h_gd > 84 && h_gd <= 90 %Section 2: Single Variation
    Rho = Rho_i(2).*((1./(1 + deltaH(2).*(h_gd - h_i(2))./RE))).^ ...
        ((1 + Constant_A(2))./Constant_A(2)));

elseif h_gd > 90 && h_gd <= 106 %Section 3: Single Variation
    Rho = Rho_i(3).*((1./(1 + deltaH(3).*(h_gd - h_i(3))./RE))).^ ...
        ((1 + Constant_A(3))./Constant_A(3)));

elseif h_gd > 106 && h_gd <= 120 %Section 4: Single Variation
    Rho = Rho_i(4).*((1./(1 + deltaH(4).*(h_gd - h_i(4))./RE))).^ ...
        ((1 + Constant_A(4))./Constant_A(4)));

elseif h_gd > 120 && h_gd <= 1000 %Section 5: Power Model
    Rho = ((4.50847623E7).*(h_gd).^(-7.44605852)).*((1000)^3);

    %Note: 'Power Model' formulated with altitude in units of (km) and
    %      the output density in (kg/m^3)

else %if h_gd > 1000;
    Rho = 0;
end

end
end

```

AtmosModel.m

```
function [Rho_Vec] = AtmosModel_PostAnalysis(h_gd,AtmosModel_Choice)

global RE BetaH Rho0

WGS84Constants; %Loads global constants from external m-file

%Note: AtmosModel_Choice
%1 = Exponential density model
%2 = Combined density model (approximation of MSIS model)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Exponential Density Model (kg/km^3)
if AtmosModel_Choice == 1
    Rho = Rho0.*exp(-BetaH.*h_gd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Combined Density Model (kg/km^3)
%Note: Exponential Model: h < 84 km
% Scale Height (v1) Variation Model: 84 <= h <= 120 km
% Power Model: 121 <= h <= 1000 km

elseif AtmosModel_Choice == 2
    %Reference altitude (km)
    h_i = [67; 85; 99; 110];

    %Reference density (kg/km^3)
    Rho_i = [1.4975e-4; 7.726e-6; 4.504e-7; 5.930e-8] * (1000)^3;

    %Reference scale height (km)
    Hi = [6.6597; 4.979; 5.905; 8.731];

    %Reference molecular scale temperature (K)
    TMi = [222.8; 165.7; 195.6; 288.2];

    %Atmospheric constant (K/km)
    Constant_A = [0.1296385; 0.1545455; 0.1189286; 0.5925240];

    %Atmospheric constant (K/km)
    Constant_B = [4.044231; 0.0; 3.878571; 19.17964];

    %Dimensionless parameters
    deltaH = (Constant_A.*RE)./Hi;
    deltaTM = (Constant_B.*RE)./TMi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Density Model Altitude Section Functions
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(h_gd);
```

```

%Altitude Sections
if h_gd(ii) <= 84 %Section 1: Exponential model
    Rho = Rho0.*exp(-BetaH.*h_gd(ii));

elseif h_gd(ii) > 84 && h_gd(ii) <= 90 %Section 2: Single Variation
    Rho = Rho_i(2).*((1./(1 + deltaH(2).*(h_gd(ii) - h_i(2))./RE))).^ ...
        ((1 + Constant_A(2))./Constant_A(2)));

elseif h_gd(ii) > 90 && h_gd(ii) <= 106 %Section 3: Single Variation
    Rho = Rho_i(3).*((1./(1 + deltaH(3).*(h_gd(ii) - h_i(3))./RE))).^ ...
        ((1 + Constant_A(3))./Constant_A(3)));

elseif h_gd(ii) > 106 && h_gd(ii) <= 120 %Section 4: Single Variation
    Rho = Rho_i(4).*((1./(1 + deltaH(4).*(h_gd(ii) - h_i(4))./RE))).^ ...
        ((1 + Constant_A(4))./Constant_A(4)));

elseif h_gd(ii) > 120 && h_gd(ii) <= 1000 %Section 5: Power Model
    Rho = ((4.50847623E7).*(h_gd(ii)).^(-7.44605852)).*((1000)^3);

    %Note: 'Power Model' formulated with altitude in units of (km) and
    % the output density in (kg/m^3)

else %if h_gd > 1000;
    Rho = 0;
end

mm = mm + 1;
Rho_Vec(mm,1) = Rho;

end

end

```


EntryEOM_Complete.m

```
function Y = EntryEOM_Complete(r,V,lon,lat,fpa,heading,bank,...
                                OmegaRot,mass,S,Cd,Cl,Thrust,epiT,zetaT)

global RE FlatE Gravity_Choice

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Variable/Function Simplification
%%Latitude (lat) and longitude (lon)
clon  = cos(lon);    slon  = sin(lon);
clat  = cos(lat);    slat  = sin(lat);    tlat = tan(lat);

%%Flight-path (fpa), heading (psi), and bank (sigma) angles
cfpa  = cos(fpa);    sfpa  = sin(fpa);    tfpa = tan(fpa);
cpsi  = cos(heading); spsi  = sin(heading);
cbank = cos(bank);    sbank = sin(bank);

%%Thrust vector angles
cepiT = cos(epiT);    sepiT = sin(epiT);
czetaT = cos(zetaT);  szetaT = sin(zetaT);

%%Thrust components of Force Equations
Thrust_V  = (Thrust/mass)*(czetaT*cepiT);
Thrust_fpa = (Thrust/mass)*(szetaT*sbank + czetaT*sepiT*cbank);
Thrust_psi = Thrust*(czetaT*sepiT*sbank - szetaT*cbank);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planetary Model
[GravModel] = GravityModel(r,lat);
g           = GravModel.g;    %Spherical gravity model (km/s^2)
g_r         = GravModel.J2g_r; %Radial component of gravity (km/s^2)
g_p         = GravModel.J2g_p; %Transverse component of gravity (km/s^2)

if Gravity_Choice == 1    %Spherical gravity model
    h_gd = r - RE;
elseif Gravity_Choice == 2    %J2 gravity model
    [h_gd, lat_gd] = Geocentric2Geodetic(r,lat,RE,FlatE);
end

%%Atmospheric density (kg/km^3)
[Rho] = AtmosModel(h_gd,2);

%%Planetary rotation parameter
OmegaRot2 = OmegaRot^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Aerodynamics
D = 0.5*Rho*Cd*S*(V^2); %Drag force (kg.km/s^2)
L = 0.5*Rho*Cl*S*(V^2); %Lift force (kg.km/s^2)
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Kinematic Equations (Hicks, 42)
%%Radial position (r) differential equation
r_dot    = V*sfpa;

%Longitude (lon) differential equation
lon_dot  = ((V*cfpa*cpsi)/(r*clat));

%Latitude (lat) differential equation
lat_dot  = (1/r)*(V*cfpa*spsi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Force Equations
if Gravity_Choice == 1 %Spherical gravity model (Hicks, 52)
    %Velocity (V) differential equation
    V_dot  = Thrust_V - (D/mass) - (g*sfpa) + ...
              (r*OmegaRot2*clat*(clat*sfpa - slat*spsi*cfpa));

    %Flight-path angle (fpa) differential equation
    fpa_dot = (1/V)*(Thrust_fpa + ((L/mass)*cbank) - (g*cfpa) + ...
                    ((V^2)/r)*cfpa + (2*V*OmegaRot*clat*cpsi) + ...
                    (r*OmegaRot2*clat*(clat*cfpa + slat*spsi*sfpa)));

    %Heading angle (psi) differential equation
    psi_dot = (1/V)*(((Thrust_psi + L*sbank)/(mass*cfpa)) - ...
                    ((V^2)/r)*(cfpa*cpsi*tlat) + ...
                    (2*V*OmegaRot)*(spsi*clat*tfpa - slat)) - ...
              ((r*OmegaRot2)/cfpa)*(slat*clat*cpsi));

elseif Gravity_Choice == 2 %J2 gravity model (Hicks, 413)
    %Velocity (V) differential equation
    V_dot  = Thrust_V - (D/mass) - (g_r*sfpa) - (g_p*sfpa*cfpa) + ...
              (r*OmegaRot2*clat*(clat*sfpa - slat*spsi*cfpa));

    %Flight-path angle (fpa) differential equation
    fpa_dot = (1/V)*(Thrust_fpa + ((L/mass)*cbank) - ...
                    (g_r*cfpa) + (g_p*(sfpa^2)) + ...
                    ((V^2)/r)*cfpa + (2*V*OmegaRot*clat*cpsi) + ...
                    (r*OmegaRot2*clat*(clat*cfpa + slat*spsi*sfpa)));

    %Heading angle (psi) differential equation
    psi_dot = (1/V)*(((Thrust_psi + L*sbank)/(mass*cfpa)) - ...
                    (cpsi/cfpa)*g_p - ((V^2)/r)*(cfpa*cpsi*tlat) + ...
                    (2*V*OmegaRot)*(spsi*clat*tfpa - slat) - ...
                    ((r*OmegaRot2)/(cfpa))*(slat*clat*cpsi));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Differential Equation Vector
Y = [r_dot; V_dot; lon_dot; lat_dot; fpa_dot; psi_dot];

```

EntryEOM_Simple.m

```
function Y = EntryEOM_Simple(r,V,lon,lat,fpa,heading,bank,mass,S,Cd,Cl)
    global RE BetaH Rho0

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Variable Simplification
    %Initial latitude (lat) and longitude (lon)
    clon = cos(lon);    slon = sin(lon);
    clat = cos(lat);    slat = sin(lat);    tlat = tan(lat);

    %Initial flight-path (fpa), heading (psi), and bank (sigma) angles
    cfpa = cos(fpa);    sfpa = sin(fpa);    cpsi = cos(heading);
    spsi = sin(heading); cbank = cos(bank); sbank = sin(bank);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Planetary Model
    [GravModel] = GravityModel(r,lat);
    g          = GravModel.g;    %Spherical gravity model (km/s^2)
    h          = r - RE;        %Altitude (km)
    [rho_r] = AtmosModel(h,2); %Atmospheric density (kg/km^3)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Vehicle Aerodynamics
    D = 0.5*rho_r*Cd*S*(V^2); %Drag force (kg.km/s^2)
    L = 0.5*rho_r*Cl*S*(V^2); %Lift force (kg.km/s^2)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Kinematic Equations (Hicks, 42)
    %Radial position (r) differential equation
    r_dot = V*sfpa;

    %Longitude (lon) differential equation
    lon_dot = ((V*cfpa*cpsi)/(r*clat));

    %Latitude (lat) differential equation
    lat_dot = (1/r)*(V*cfpa*spsi);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Force Equations (Hicks, 52)
    %Velocity (V) differential equation
    V_dot = -(D/mass) - (g*sfpa);

    %Flight-path angle (fpa) differential equation
    fpa_dot = (1/V)*((L/mass)*cbank - (g*cfpa) + ((V^2)/r)*cfpa);

    %Heading angle (psi) differential equation
    psi_dot = (1/V)*(((L*sbank)/(mass*cfpa)) - ((V^2)/r)*(cfpa*cpsi*tlat));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Differential Equation Vector
    Y = [r_dot; V_dot; lon_dot; lat_dot; fpa_dot; psi_dot];
```

EntryEOM_Euler.m

```
function [T_total, traj_param] = EntryEOM_Euler(r,V,lon,lat,fpa,heading,...
                                              bank,deltaT,Time_max)

global g0 RE OmegaE BetaH Rho0 Vehicle_Choice Omega_Choice BankAngle_Choice

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Maneuver Profile Angles (rad)
lon(1)      = lon;      %Longitude
lat(1)      = lat;      %Latitude
fpa(1)      = fpa;      %Flight-path angle
heading(1)  = heading;  %Heading angle
bank(1)     = bank;     %Bank angle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planetary Model
[GravModel] = GravityModel(r,lat);
g          = GravModel.g;      %Spherical gravity model (km/s^2)
rho_r      = Rho0*exp(-BetaH*(r - RE)); %Atmospheric density (kg/km^3)

if          Omega_Choice == 1
    OmegaRot = OmegaE; %Planetary rotation "activated"
elseif Omega_Choice == 2
    OmegaRot = 0;      %Planetary rotation "de-activated"
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Variable/Function Simplification
OmegaRot2 = (OmegaRot)^2; %Planetary rotation parameter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
[Vehicle] = VehicleSpecs(Vehicle_Choice);

mass = Vehicle.mass;      %Mass (kg)
S_m2 = Vehicle.S_m2;      %Planform area (m^2)
S     = S_m2/(1000^2);    %Planform area (km^2)
Cd    = Vehicle.Cd;       %Drag coefficient
Cl    = Vehicle.Cl;       %Lift coefficient

D     = 0.5*rho_r*Cd*S*(V^2); %Drag force (kg.km/s^2)
L     = 0.5*rho_r*Cl*S*(V^2); %Lift force (kg.km/s^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Numerical Integration of Equations of Motion
r(1)   = r;   V(1)   = V;      %Initial conditions for vehicle dynamics
g(1)   = g;   rho_r(1) = rho_r; %Initial conditions for entry environment
D(1)   = D;   L(1)   = L;      %Initial conditions for vehicle aerodynamics
```

```

%Initial vehicle deceleration
a_decel_v(1) = (D(1)/mass) + g(1)*sin(fpa(1));
a_decel_L(1) = (-L(1)/mass) - (((V(1)^2)/r(1)) - g(1))*cos(fpa(1));
a_decel_mag(1) = sqrt((a_decel_v(1))^2 + (a_decel_L(1))^2);
ag_decel_mag(1) = a_decel_mag(1)/g(1);

%Initial vehicle stagnation and wall heat flux
qdot_s(1) = sqrt((rho_r(1)*S*Cd)/(2*mass*BetaH))* ...
            ((V(1)^2)/(2*g(1)*r(1)))^(3/2);
qdot_w(1) = ((rho_r(1)*S*Cd)/(2*mass*BetaH))* ...
            ((V(1)^2)/(2*g(1)*r(1)))^(3/2);

T_total(1) = 0; %Initial condition for total mission time
ctr = 1; %Iteration counter initiation

while (T_total <= Time_max)
    %% Kinematic Equations
    %Radial position (r) differential equation
    r_dot = V(ctr)*sin(fpa(ctr));

    %Longitude (lon) differential equation
    lon_dot = ((V(ctr)*cos(fpa(ctr))*cos(heading(ctr)))/ ...
                (r(ctr)*cos(lat(ctr))));

    %Latitude (lat) differential equation
    lat_dot = (V(ctr)*cos(fpa(ctr))*sin(heading(ctr)))/r(ctr);

    %% Force Equations
    %Velocity (V) differential equation
    V_dot = -(D(ctr)/mass) - (g(ctr)*sin(fpa(ctr))) + ...
              (r(ctr)*OmegaRot2*cos(lat(ctr))* ...
              (cos(lat(ctr))*sin(fpa(ctr)) - ...
              sin(lat(ctr))*sin(heading(ctr))*cos(fpa(ctr))));

    %Flight-path angle (gamma) differential equation
    Vgamma_dot = ((L(ctr)*cos(bank(ctr)))/mass)-(g(ctr)*cos(fpa(ctr))) + ...
                  ((V(ctr)^2)/r(ctr))*cos(fpa(ctr)) + ...
                  (2*V(ctr)*OmegaRot*cos(lat(ctr))*cos(heading(ctr))) + ...
                  (r(ctr)*OmegaRot2*cos(lat(ctr))* ...
                  (cos(lat(ctr))*cos(fpa(ctr)) + ...
                  sin(lat(ctr))*sin(heading(ctr))*sin(fpa(ctr))));

    %Heading angle (psi) differential equation
    Vpsi_dot = ((L(ctr)*sin(bank(ctr)))/(mass*cos(fpa(ctr)))) - ...
                (((V(ctr)^2)/r(ctr))*cos(fpa(ctr))* ...
                cos(heading(ctr))*tan(lat(ctr))) + ((2*V(ctr)*OmegaRot)* ...
                (sin(heading(ctr))*cos(lat(ctr))*tan(fpa(ctr)) - ...
                sin(lat(ctr)))) - (((r(ctr)*OmegaRot2)/(cos(fpa(ctr))))* ...
                (sin(lat(ctr))*cos(lat(ctr))*cos(heading(ctr))));

    %% Parameter Updates
    %Updates to Vehicle Dynamics
    r(ctr+1) = r(ctr) + r_dot*deltaT; %Radial position
    V(ctr+1) = V(ctr) + V_dot*deltaT; %Velocity

```

```

%Updates to Maneuver Profile Angles
lon(ctr+1)      = lon(ctr) + lon_dot*deltaT;           %Longitude
lat(ctr+1)      = lat(ctr) + lat_dot*deltaT;           %Latitude
fpa(ctr+1)      = fpa(ctr) + (Vgamma_dot/V(ctr))*deltaT; %Flight-path angle
heading(ctr+1)  = heading(ctr)+(Vpsi_dot/V(ctr))*deltaT; %Heading angle

%Updates to Simulation Environment
g(ctr+1)        = g0*((RE/r(ctr+1))^2);                %Grav. acceleration
rho_r(ctr+1)    = Rho0*exp(-BetaH*(r(ctr+1) - RE));    %Density
D(ctr+1)        = 0.5*rho_r(ctr+1)*Cd*S*(V(ctr+1)^2);  %Drag force
L(ctr+1)        = 0.5*rho_r(ctr+1)*Cl*S*(V(ctr+1)^2);  %Lift force
T_total(ctr+1)  = T_total(ctr) + deltaT;                %Trajectory time

if      BankAngle_Choice == 1
    bank(ctr+1) = bank(ctr);

elseif BankAngle_Choice == 2
    load Apollo_10_BankAngle; %Loads bank angle profile
    BankAngle_time = Apollo_10_BankAngle(:,1);
    BankAngle_rad  = deg2rad(Apollo_10_BankAngle(:,2));
    bank(ctr+1) = interp1(BankAngle_time,BankAngle_rad,T_total(ctr+1));
end

ctr = ctr + 1; %Update to iteration counter
end

%Trajectory solution vectors
T_total = T_total';
traj_param = [r' V' lon' lat' fpa' heading' bank'];

```

GravityModel.m

```
function [GravModel] = GravityModel(r,phi_gc)

global MU g0 RE J2 J3 J4

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lat  = phi_gc; %Geocentric latitude (rad)
clat = cos(lat); slat = sin(lat); %Variable simplification
RE_r = RE./r; %Ratio of planet radius/radius of interest

%Legendre polynomials
P0 = 1;
P1 = slat;
P2 = (1/2).*(3. *(slat.^2) - 1);
P3 = (1/2).*(5. *(slat.^3) - 3.*slat);
P4 = (1/8).*(35.*(slat.^4) - 30.*(slat.^2) + 3);
P5 = (1/8).*(63.*(slat.^5) - 70.*(slat.^3) + 15.*slat);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Spherical (Newtonian) Gravity Model (km/s^2)
GravModel.g = g0.*((RE./r).^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% J2-Gravity Model (km/s^2)
%Reverse radial direction along unit vector toward planetary center
GravModel.J2g_r = (MU./r.^2).*(1 - 3.*J2.*(RE_r.^2).*P2);

%Reverse transverse direction along unit vector toward planetary north
GravModel.J2g_p = ((3.*MU)./r.^2).*(RE_r.^2).*clat.*slat.*J2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% J4-Gravity Model (km/s^2)
%Reverse radial direction along unit vector toward planetary center
GravModel.J4g_r = (MU./r.^2).*(1 - 3.*J2.*(RE_r.^2).*P2 - ...
    4.*J3.*(RE_r.^3).*P3 - ...
    5.*J4.*(RE_r.^4).*P4);

%Reverse transverse direction along unit vector toward planetary north
GravModel.J4g_p = ((3.*MU)./r.^2).*(RE_r.^2).*clat.*slat.*
    (J2 + (1/2).*J3.*(RE_r).*(1./slat).*(5.*(slat.^2)-1) + ...
    (5/6).*J4.*(RE_r.^2).*(7.*(slat.^2)-1));
```

VehicleSpecs.m

```
function [Vehicle] = VehicleSpecs(Vehicle_Choice)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 'Vehicle_Choice' Options
%1 = Notional Transatmospheric Vehicle (TAV)
%2 = ESPA Secondary Payload Notional Satellite
%3 = Primary Payload Notional Satellite
%4 = Apollo 10 Command Module Capsule
%5 = Apollo 10 CM Capsule w/ Alternative Cl/Cd
%6 = Notional Satellite

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Spacecraft Data
L2D = linspace(0.8,2.0,25); %Lift-to-drag ratio vector

%L2D = [0.80,0.85,0.90,0.95,1.00,
%       1.05,1.10,1.15,1.20,1.25,
%       1.30,1.35,1.40,1.45,1.50,
%       1.55,1.60,1.65,1.70,1.75,
%       1.80,1.85,1.90,1.95,2.00]

if Vehicle_Choice == 1 %Notional Trans-Atmospheric Vehicle (TAV)
    Vehicle.mass = 5000;           %Wet mass (kg)
    Vehicle.S_m2 = 18;            %Planform area (m^2)
    Vehicle.Cd = 0.5;             %Drag coefficient
    Vehicle.Cl = 3.0;             %Lift coefficient
    Vehicle.Rn = 0.3048;          %Nose radius (m)

    %Notional Trans-Atmospheric Vehicle (TAV)
    % Vehicle.mass = 4000;          %Wet mass (kg)
    % Vehicle.S_m2 = 10;            %Planform area (m^2)
    % Vehicle.Cd = 1.0;            %Drag coefficient
    % Vehicle.Cl = 6.6;            %Lift coefficient

    %X-37B Lifting Entry Vehicle
    % Vehicle.mass = 4989.5;        %Wet mass (kg)
    % Vehicle.S_m2 = 18.63;        %Planform area (m^2)
    % Vehicle.Cd = 0.5;            %Drag coefficient
    % Vehicle.Cl = L2D(5).*Vehicle.Cd; %Lift coefficient

elseif Vehicle_Choice == 2 %ESPA SPL Notional Satellite
    Vehicle.mass = 200;           %Wet mass (kg)
    Vehicle.S_m2 = 18.63;         %Planform area (m^2)
    Vehicle.Cd = 2.2;            %Drag coefficient
    Vehicle.Cl = L2D(5).*Vehicle.Cd; %Lift coefficient

elseif Vehicle_Choice == 3 %Primary Payload Notional Satellite
    Vehicle.mass = 1000;          %Wet mass (kg)
    Vehicle.S_m2 = 18.63;         %Planform area (m^2)
    Vehicle.Cd = 2.2;            %Drag coefficient
    Vehicle.Cl = L2D(5).*Vehicle.Cd; %Lift coefficient
```



```

elseif Vehicle_Choice == 4 %Apollo 10 Command Module (CM) Capsule
    Vehicle.mass = 5498.22;           %Mass (kg)
    Vehicle.S_m2 = 12.017;           %Planform area (m^2)
    Vehicle.Cd    = 1.2569;          %Drag coefficient
    Vehicle.Cl    = 0.4082;          %Lift coefficient

elseif Vehicle_Choice == 5 %Apollo 10 CM Capsule w/ Alternative Cl/Cd
    Vehicle.mass = 5498.22;           %Mass (kg)
    Vehicle.S_m2 = 12.017;           %Planform area (m^2)
    Vehicle.Cd    = 1.255;           %Drag coefficient
    Vehicle.Cl    = 0.4225;          %Lift coefficient

elseif Vehicle_Choice == 6 %Notional Satellite
    Vehicle.mass = 2000;              %Mass (kg)
    Vehicle.S_m2 = 10;               %Planform area (m^2)
    Vehicle.Cd    = 3.0;             %Drag coefficient
    Vehicle.Cl    = 0;              %Lift coefficient

elseif Vehicle_Choice == 9 %TAV from DOE
    Vehicle.mass = 2000;              %Mass (kg)
    Vehicle.S_m2 = 18.5;             %Planform area (m^2)
    Vehicle.Cd    = 0.5;             %Drag coefficient
    Vehicle.Cl    = 3.0;             %Lift coefficient
end

%Engine Parameters
%Max Thrust Options: 14679 N (Impulsive thrusting; H2O2/JP-8; X-37B)
%                      13345 N (Impulsive thrusting; H2O2/JP-10; X-37B)
%                      9901 N (Impulsive thrusting; H2O2; X-37B)
%                      300E-3 N (Continuous thrusting; notional satellite)
%                      500E-3 N (Continuous thrusting; notional satellite)

Vehicle.T_Max    = 0 * (1/1000); %Maximum thrust, (N)->(kg.km/s^2)
Vehicle.Throttle = 100;          %Throttle (percentage)

%Magnitude of thrust (kg.km/s^2)
Vehicle.Thrust    = Vehicle.T_Max * (Vehicle.Throttle/100);

%Angle describing projection of thrust vector on L-V plane (rad)
Vehicle.epsT      = deg2rad(0.0);

%Angle describing projection of thrust vector on R-V plane (rad)
Vehicle.zetaT     = deg2rad(0.0);

```

WGS84Constants.m

```
function WGS84Constants

global MU g0 RE OmegaE J2 J3 J4 J6 FlateE EccE BetaH Rho0 BR StefBoltz Boltz

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Earth Planetary Constants
MU      = 398600.442;    %Gravitational parameter (km^3/s^2)
RE      = 6378.137;     %Planetary radius (km)
g0      = MU/(RE^2);    %Sea-level gravitational acceleration (km/s^2)
OmegaE  = 7.2921158e-5; %Planetary rotational velocity (rad/s)

%Jeffery's Constants
J2      = 0.0010826269;
J3      = -0.0000025323;
J4      = -0.0000016204;
J6      = -0.0000021;

%Planetary Eccentricity Calculation
FlateE  = 1.0/298.257;    %Flattening parameter (f)
EccE2   = (2.0 - FlateE)*FlateE; %Square of planetary eccentricity
EccE    = sqrt(EccE2);    %Planetary eccentricity

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Earth Atmospheric Constants
BetaH   = 0.14;          %Atmospheric scale height (km^-1)
Rho0    = 1.225 * (1000)^3; %Atmospheric density @ planetary surface (kg/km^3)
BR      = 900;           %Average parameter for universal formulation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Physical Constants
StefBoltz = 5.67E-8;      %Stefan-Boltzmann constant (W.m^-2.K^-4)
Boltz     = 1.380658E-23; %Boltzmann constant (J/K)
```

Appendix F. MATLAB® Code for Maneuver Simulations

Table F.1. m-File Classification for Maneuver Simulations

Filename	File Type	Description
BankManeuvers	Script	Skip entry maneuvers
BankManeuvers_Function	Function	Skip entry maneuver function
BankManeuvers_fxnDOE	Function	Skip entry for DOE
BankManeuvers_fxnDOE_Hohmann	Function	Skip entry for DOE with Hohmann
BankManeuvers_MultiAOT	Function	Skip entry and descent-boost
BiElliptic	Function	Bi-elliptic transfer
BiElliptic_VelInput	Function	Bi-elliptic transfer, V specified
DescentBoost_Molniya	Script	Descent-boost Molniya injection
DescentBoost_ReCirc	Script	Descent-boost orbit injection
Hohmann_Analysis_Molniya	Script	Hohmann for Molniya injection
Hohmann_Combined	Function	Combined Hohmann transfer
Hohmann_Combined_dI	Function	Combined Hohmann, Δi specified
Hohmann_Combined_VelInput	Function	Combined Hohmann, V specified
Hohmann_Geocentric	Function	Hohmann, geocentric coordinates
Hohmann_Geodetic	Function	Hohmann, geodetic coordinates
Hohmann_SkipReCirc	Function	Hohmann at skip apogee
Hohmann_VelInput	Function	Hohmann, V specified
PlanarManeuvers	Function	Planar phasing maneuvers
RefOrb_Targeting	Script	Simple plane change maneuvers
Trajectory_3DPlotting	Script	Three-dimensional plotting

BankManeuvers.m

```
clear all; clc; close all;
```

```
global MU RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape
```

```
WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Selection and Targeting Loop Initialization
LatJump_Change = 1;
LonThreshold    = 35;
LatThreshold    = 35;
Target_Choice   = 2;
Vehicle_Choice  = 9;
Map_Choice      = 1;

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver;    Lon_Target = Lon_Denver;    dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow;   Lon_Target = Lon_Glasgow;   dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny;    Lon_Target = Lon_Grozny;    dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow;    Lon_Target = Lon_Moscow;    dLat = 3;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti;     Lon_Target = Lon_Ponti;     dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong;     Lon_Target = Lon_Pyong;     dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk;      Lon_Target = Lon_Reyk;      dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran;    Lon_Target = Lon_Tehran;    dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo;     Lon_Target = Lon_Tokyo;     dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil;    Lon_Target = Lon_Brasil;    dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos;    Lon_Target = Lon_Buenos;    dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra;  Lon_Target = Lon_Canberra;  dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape;      Lon_Target = Lon_Cape;      dLat = 4;
end

h_Perig0 = 87;
h_Perig  = h_Perig0;
% MissDistance = 9999;
% WhileCount   = 0;
% while MissDistance > 1.0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 9 %VEHICLE SELECTION OVERRIDE
    mass = 2000; %Mass (kg)
    S_m2 = 18.5; %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = 0.5; %Drag coefficient
    Cl    = 3.0; %Lift coefficient
else

```

```

[Vehicle] = VehicleSpecs(Vehicle_Choice);
mass = Vehicle.mass; %Mass (kg)
S_m2 = Vehicle.S_m2; %Planform area (m^2)
S = S_m2/(1000^2); %Planform area (km^2)
Cd = Vehicle.Cd; %Drag coefficient
Cl = Vehicle.Cl; %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
Time_Max = 1.00; %Maximum simulation time (days)
ecc_Ref = 0; %Orbit eccentricity
h_Apog = 1000; %Apogee altitude (km)
lon_Ref = 0; %Initial longitude (deg)
lat_Ref = 0; %Initial geodetic latitude (deg)
fpa_Ref = 0; %Flight-path angle (deg)
PSI_Ref = 37.835; %Heading angle (deg)
bank_Ref = 0; %Reference orbit bank angle (deg)
bank_Skip = -90; %Skip maneuver bank angle (deg)

%Converts and overwrites initial angle variables
lon_Ref = deg2rad(lon_Ref); lat_Ref = deg2rad(lat_Ref);
fpa_Ref = deg2rad(fpa_Ref); PSI_Ref = deg2rad(PSI_Ref);

%Reference orbit parameters
r_Apog = h_Apog + RE; %Apogee radial position (km)
r_Perig = h_Perig + RE; %Perigee radial position (km)
SMA_Ref = 0.5*(r_Apog + r_Apog); %Reference orbit semi-major axis (km)
SMA_Skip = 0.5*(r_Apog + r_Perig); %Skip orbit semi-major axis (km)
RefPeriod = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_Apog,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank_Ref);

%Apogee velocity for non-rotating frame (km/s)
V_Apog = sqrt((2*MU*r_Perig)/(r_Apog*(r_Apog + r_Perig)));

%Conversion of time units from days to seconds
Time_Max = Time_Max*(24)*(60)*(60);

SMA_Target0 = SMA_Ref; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel; %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0, Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
                                                1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(1,1), ...
                                                lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank_Ref);

```

```

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_Apog + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
    ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
    ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0, Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(ii,1), ...
            lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank_Ref);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_Apog + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
            (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
            ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));

        %Updated heading angle (rad)
        PSI_Check0(ii+1,1) = PSI_Ref + ...
            asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));

```

```

IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
                                PSI_Check0(ii-1,1));

ii = ii + 1; %Update to row-index counter
IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t, RefOrb_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Apog,V_Rel0, ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank_Ref);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Descent Velocity
IterMax = 20; %Maximum number of iterations
SMA0 = SMA_Ref; %Initial guess for semi-major axis (km)
SMA_Target = SMA_Skip; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check(1,1) = .96*V_Apog; %Initial guess for descent velocity (km/s)
% Set 'V_Check' coeff. to 0.98 for over-flights; 0.96 for max. inclination
PSI_Check(1,1) = PSI_Ref; %Initial guess for heading angle (rad)

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[t_vec, traj_states] = Maneuver_MainFunction(Vehicle_Choice,1,1,1,1,1, ...
    0.5*RefPeriod,r_Apog,V_Check(1,1),lon_Ref, ...
    lat_Ref,fpa_Ref,PSI_Check(1,1),bank_Skip);

%Perigee radial position (km)
[r_Check(1,1),Perig_Index] = min(traj_states(:,1));
%Semi-major axis (km)
SMA_Check(1,1) = 0.5*(r_Apog + r_Check(1,1));

%Iteration error (s)
GuessError(1,1) = -((SMA_Check(1,1) - SMA0)/ ...
    ((V_Check(1,1) - V_Decrement) - V_Check(1,1)));

%Updated velocity (km/s)
V_Check(2,1) = (V_Check(1,1) - V_Decrement) - ...
    ((SMA_Target - SMA_Check(1,1))/GuessError(1,1));

%Updated heading angle (rad)
PSI_Check(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA(1,1) = abs(SMA_Target - SMA_Check(1,1));
IterativeDiff_PSI(1,1) = abs(PSI_Check(2,1) - PSI_Check(1,1));
IterCount = 1; %Initialization of iteration counter for Secant loop

```

```

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target - SMA_Check(ii-1,1)) > 1E-6 && ...
        abs(PSI_Check(ii,1) - PSI_Check(ii-1,1)) > 1E-8 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [t_vec, traj_states] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*RefPeriod,r_Apog, ...
            V_Check(ii,1),lon_Ref,lat_Ref,fpa_Ref, ...
            PSI_Check(ii,1),bank_Skip);

        %Current iteration perigee radial position (km)
        [r_Check(ii,1),Perig_Index] = min(traj_states(:,1));
        %Current iteration semi-major axis (km)
        SMA_Check(ii,1) = 0.5*(r_Apog + r_Check(ii,1));
        %Iteration error (sec)
        GuessError(ii,1) = -((SMA_Check(ii,1) - SMA_Check(ii-1,1))/ ...
            (V_Check(ii,1) - V_Check(ii-1,1)));

        %Updated velocity (km/s)
        V_Check(ii+1,1) = V_Check(ii,1) - ...
            ((SMA_Target - SMA_Check(ii,1))/GuessError(ii,1));

        %Updated heading angle (rad)
        PSI_Check(ii+1,1) = PSI_Ref + ...
            asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA(ii,1) = abs(SMA_Target - SMA_Check(ii,1));
        IterativeDiff_PSI(ii,1) = abs(PSI_Check(ii,1) - PSI_Check(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

V_Maneuver = V_Check(ii,1); %Descent velocity for target SMA
dV_Maneuver = abs(V_Maneuver - V_Rel); %Maneuver delta-V (km/s)
PSI_Rel = rad2deg(PSI_Check(ii,1)); %Relative heading angle (deg)

%Trajectory simulation for skip maneuver
[Skip_t, Skip_States] = SingleSkip_Maneuver(Vehicle_Choice,1,1,1,1,1, ...
    RefPeriod,r_Apog,V_Maneuver,lon_Ref, ...
    lat_Ref,fpa_Ref,deg2rad(PSI_Rel),bank_Skip);

Perigee_Altitude = min(Skip_States(:,1)) - RE

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Propagation of Re-Circularized Orbit
Time_Max = 8000; %Maximum simulation time (s)
ecc = 0; %Orbit eccentricity
r_Prop = Skip_States(end,1); %Orbit radial position (km)
h_Prop = r_Prop - RE; %Orbit altitude (km)
lon_Prop = Skip_States(end,3); %Initial longitude (rad)
lat_Prop = Skip_States(end,4); %Initial geodetic latitude (rad)
fpa_Prop = 0; %Flight-path angle (rad)
PSI_Prop = -(min(Skip_States(:,4))); %Heading angle (rad)
bank_Prop = bank_Skip; %Bank angle (deg)

%Re-circularized orbit parameters
SMA_Prop = 0.5*(r_Prop + r_Prop); %Semi-major axis (km)
Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_RelProp,PSI_RelProp] = RelativeStates(mass,S,Cd,Cl,h_Prop,lat_Prop, ...
                                         fpa_Prop,PSI_Prop,bank_Prop);

SMA_TargetProp = SMA_Prop;
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckP(1,1) = V_RelProp; %Initial guess for velocity (km/s)
PSI_CheckP(1,1) = PSI_RelProp; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
                                                1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
                                                V_CheckP(1,1),lon_Prop,lat_Prop, ...
                                                fpa_Prop,PSI_CheckP(1,1),bank_Prop);

[r_CheckP(1,1),ApogFlag] = min(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckP(1,1) = 0.5*(r_Prop + r_CheckP(1,1));

%Iteration error (s)
GuessErrorP(1,1) = -((SMA_CheckP(1,1) - SMA_Prop)/ ...
                    ((V_CheckP(1,1) - V_Decrement) - V_CheckP(1,1)));

%Updated velocity (km/s)
V_CheckP(2,1) = (V_CheckP(1,1) - V_Decrement) - ...
                ((SMA_TargetProp - SMA_CheckP(1,1))/GuessErrorP(1,1));

%Updated heading angle (rad)
PSI_CheckP(2,1) = PSI_Prop + ...
                asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckP(1,1));
IterativeDiff_PSI_Prop(1,1) = abs(PSI_CheckP(2,1) - PSI_CheckP(1,1));
IterCount = 1; %Initializes iteration counter for Secant loop

```

```

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckP(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckP(ii,1) - PSI_CheckP(ii-1,1)) > 1E-10 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckP(ii,1), ...
            lon_Prop,lat_Prop,fpa_Prop, ...
            PSI_CheckP(ii,1),bank_Prop);

        [r_CheckP(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

        %Current iteration semi-major axis (km)
        SMA_CheckP(ii,1) = 0.5*(r_Prop + r_CheckP(ii,1));

        %Iteration error (sec)
        GuessErrorP(ii,1) = -((SMA_CheckP(ii,1) - SMA_CheckP(ii-1,1))/ ...
            (V_CheckP(ii,1) - V_CheckP(ii-1,1)));

        %Updated velocity (km/s)
        V_CheckP(ii+1,1) = V_CheckP(ii,1) - ...
            ((SMA_TargetProp - SMA_CheckP(ii,1))/GuessErrorP(ii,1));

        %Updated heading angle (rad)
        PSI_CheckP(ii+1,1) = PSI_Prop + ...
            asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp-SMA_CheckP(ii,1));
        IterativeDiff_PSI_Prop(ii,1) = abs(PSI_CheckP(ii,1) - ...
            PSI_CheckP(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t, Orbit_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Prop,V_CheckP(ii), ...
    lon_Prop,lat_Prop,fpa_Prop, ...
    PSI_CheckP(ii),bank_Prop);

%Re-defined propagated orbit states
PropOrb_t = [Skip_t ; Skip_t(end) + Orbit_t(2:end)];
PropOrb_States = [Skip_States(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h = PropOrb_States(:,1) - RE; %Altitude (km)
PropOrb_V = PropOrb_States(:,2); %Velocity (km/s)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
    + 180),360) - 180; %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Propagated Trajectory Crossings of Target Coordinates
% %Longitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(PropOrb_Lon_deg)
%     mm = mm + 1;
%     if abs(PropOrb_Lon_deg(ii) - Lon_Target) < LonThreshold
%         LonTGT_Crossing(mm,1) = PropOrb_t(ii);
%         LonTGT_Crossing(mm,2) = PropOrb_h(ii);
%         LonTGT_Crossing(mm,3) = PropOrb_Lat_deg(ii);
%         LonTGT_Crossing(mm,4) = PropOrb_Lon_deg(ii);
%     else
%         LonTGT_Crossing(mm,1:4) = 0;
%     end
% end
%
% %Latitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(PropOrb_Lat_deg)
%     mm = mm + 1;
%     if abs(PropOrb_Lat_deg(ii) - Lat_Target) < LatThreshold
%         %% abs(PropOrb_Lon_deg(ii) - Lon_Target) < 30
%         LatTGT_Crossing(mm,1) = PropOrb_t(ii);
%         LatTGT_Crossing(mm,2) = PropOrb_h(ii);
%         LatTGT_Crossing(mm,3) = PropOrb_Lon_deg(ii);
%         LatTGT_Crossing(mm,4) = PropOrb_Lat_deg(ii);
%     else
%         LatTGT_Crossing(mm,1:4) = 0;
%     end
% end
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Determination of Indices Corresponding to Crossings
% %Longitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(LonTGT_Crossing)
%     if LonTGT_Crossing(ii) ~= 0
%         mm = mm + 1;
%         FlagVector_Lon(mm,1) = ii;
%         WithinIdent_Lon(mm,:) = LonTGT_Crossing(ii,:);
%     end
% end
% FlagVector_Lon = [FlagVector_Lon;0];
%
% %Latitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(LatTGT_Crossing)
%     if LatTGT_Crossing(ii) ~= 0
%         mm = mm + 1;
%         FlagVector_Lat(mm,1) = ii;
%         WithinIdent_Lat(mm,:) = LatTGT_Crossing(ii,:);
%     end
% end
% FlagVector_Lat = [FlagVector_Lat;0];
%

```

```

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Determination of Indices Corresponding to Jumps in Crossings
% %Longitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(FlagVector_Lon)-1
%     if abs((FlagVector_Lon(ii+1) - FlagVector_Lon(ii))) > 1
%         mm = mm + 1;
%         LonTGT_Jump(mm,1) = ii;
%     end
% end
% LonTGT_Jump = [0;LonTGT_Jump];
%
% %Latitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(FlagVector_Lat)-1
%     if abs((FlagVector_Lat(ii+1) - FlagVector_Lat(ii))) > 1
%         mm = mm + 1;
%         LatTGT_Jump(mm,1) = ii;
%     end
% end
%
% if LatJump_Change == 1 %Appropriate for 'mid-' to 'high-' latitudes
%     LatTGT_Jump = [0; LatTGT_Jump];
% elseif LatJump_Change == 2 %Appropriate for 'low-' latitudes
%     LatTGT_Jump = [LatTGT_Jump];
% end
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Interpolation of Crossing Trajectories
% %Longitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 2:length(LonTGT_Jump)
%     mm = mm + 1;
%     LonTGT_Interp(mm,:) = ...
%         interp1(LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
%                 FlagVector_Lon(LonTGT_Jump(ii)),4), ...
%                 LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
%                 FlagVector_Lon(LonTGT_Jump(ii)),1:3), ...
%                 Lon_Target,'spline'); %Cubic spline interpolation
% end
%
% %Latitude crossings
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 2:length(LatTGT_Jump)
%     mm = mm + 1;
%     LatTGT_Interp(mm,:) = ...
%         interp1(LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
%                 FlagVector_Lat(LatTGT_Jump(ii)),4), ...
%                 LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
%                 FlagVector_Lat(LatTGT_Jump(ii)),1:3), ...
%                 Lat_Target,'spline'); %Cubic spline interpolation
% end
%
% %Removal of negative perturbed periods
% LatTGT_Interp(any(LatTGT_Interp(:,1)<0,2),:) = [];
%

```

```

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% Determination of Minimum Target Miss Distance
% %Target miss distance for both spherical and oblate planetary models
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(PropOrb_Lon_deg)
%     mm = mm + 1;
%     SphereDist_Lon = CoordDist(Lon_Target,Lon_Target, ...
%                               Lat_Target,LonTGT_Interp(:,3),1);
% end
%
% %Longitudinal target miss distance (km)
% [MinDistance_Lon,MinFlag_Lon] = min(SphereDist_Lon(:,1));
%
% mm = 0; %Initializes vector concatenation counter at zero
% for ii = 1:length(PropOrb_Lat_deg)
%     mm = mm + 1;
%     SphereDist_Lat = CoordDist(Lon_Target,LatTGT_Interp(:,3), ...
%                               Lat_Target,Lat_Target,1);
% end
%
% %Latitudinal target miss distance (km)
% [MinDistance_Lat,MinFlag_Lat] = min(SphereDist_Lat(:,1));
%
% MinDist_Vec = [MinDistance_Lon, MinDistance_Lat];
% MinFlag_Vec = [MinFlag_Lon,      MinFlag_Lat];
% [MinDistance, MinIndex] = min(MinDist_Vec);
% MinFlag       = MinFlag_Vec(MinIndex);
% MissDistance = MinDistance
%
% if      MinIndex == 1
%     MinInterp = LonTGT_Interp;
% elseif MinIndex == 2
%     MinInterp = LatTGT_Interp;
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Over-Flight Parameters
%Time-of-arrival at target (hr)
% TimeArrival = (MinInterp(MinFlag,1))*(1/60)*(1/60);
%
% %Altitude-of-arrival at target (km)
% AltArrival  = MinInterp(MinFlag,2);
%
% %Payload imager field-of-view (FOV) and resolution during over-flight
% %Visible spectrum imager
% [FOV_m2_Vis, FOV_km2_Vis, Resolution_Vis] = ...
%     PayloadImager(AltArrival*(1.0E3),1.15,2.70,1.0E-6);
%
% %Latitude-of-arrival at target (deg)
% LatArrival  = LonTGT_Interp(MinFlag_Lon,3);
%
% %Longitude-of-arrival at target (deg)
% LonArrival  = LatTGT_Interp(MinFlag_Lat,3);
%
% %Maximum inclination (deg)
% MaxIncl     = max(PropOrb_Lat_deg);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Atmospheric-Transit Parameters
%Identifies time/altitude segment of trajectory within atmosphere
jj = 0; %Initializes vector concatenation counter at zero
PropOrb_th = [PropOrb_t,PropOrb_h]; %Concatenation of t, h vectors

for ii = 1:length(PropOrb_th(:,1))
    if PropOrb_th(ii,2) < 120
        jj = jj + 1;
        t_Atmos(jj,:) = PropOrb_th(ii,1); %Time (s)
        h_Atmos(jj,:) = PropOrb_th(ii,2); %Altitude (km)
    end
end

t_EnterAtmos = t_Atmos(1); %Time of atmospheric entry (h < 120 km)
t_ExitAtmos = t_Atmos(end); %Time of atmospheric exit (h > 120 km)
t_Transit = [t_EnterAtmos,t_ExitAtmos];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
[decel] = EntryDecel(1,mass,S,Cd,C1,
                    PropOrb_States(:,1),PropOrb_States(:,2), ...
                    PropOrb_States(:,4),PropOrb_States(:,5));

%Tangential deceleration (g's)
TangDecelG_Max = max(decel.TangG); %Maximum value
TangDecelG_Min = min(decel.TangG); %Minimum value

%Normal deceleration (g's)
NormDecelG_Max = max(decel.NormalG); %Maximum value
NormDecelG_Min = min(decel.NormalG); %Minimum value

%Deceleration magnitude (g's)
MagDecelG_Max = max(decel.Gs); %Maximum value
MagDecelG_Min = min(decel.Gs); %Minimum value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Heat Flux
%Atmospheric density (kg/km^3)
[PropOrb_Rho] = AtmosModel_PostAnalysis(PropOrb_h,2);

%Maximum velocity (km/s) -- Occurs at Perigee
VMax = max(PropOrb_V);

Emissivity = 0.8; %Emissivity
Tw_F = 0; %Wall temperature (deg F)
TMaxF = 1800; %Free-stream temperature (deg F)

%Heat transfer models
[HeatModel,Eta,T_KE] = HeatFluxModel(PropOrb_V,PropOrb_Rho,Emissivity, ...
                                     Tw_F,TMaxF,mass,S,Cd,C1);

```

```

%Average wall heat flux (non-dimensional)
Qw          = HeatModel.Qw;
Qw_Max      = max(Qw);    %Maximum value

%Average stagnation heat flux (non-dimensional)
Qs          = HeatModel.Qs;
Qs_Max      = max(Qs);    %Maximum value

%Stagnation heat flux (kW/m^2); Source: Rao (2002)
Qdot        = HeatModel.Qdot;
Qdot_Max    = max(Qdot); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Havey (1982)
QHavey      = HeatModel.QHavey;
QHavey_Max  = max(QHavey);

%Stagnation heat flux (kW/m^2); Source: Galman (1961)
QGalman     = HeatModel.QGalman;
QGalman_Max = max(QGalman);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %Updated perigee altitude (km)
% if      MinIndex == 1
%     if      LatArrival > Lat_Target
%         if      MissDistance > 100
%             h_Perig = h_Perig + 2.0;
%         elseif MissDistance > 20 && MissDistance <= 100
%             h_Perig = h_Perig + 1.0;
%         elseif MissDistance > 15 && MissDistance <= 20
%             h_Perig = h_Perig + 0.1;
%         elseif MissDistance > 5  && MissDistance <= 15
%             h_Perig = h_Perig + 0.01;
%         elseif MissDistance <= 5
%             h_Perig = h_Perig + 0.001;
%         end
%     %
%     elseif LatArrival < Lat_Target
%         if      MissDistance > 100
%             h_Perig = h_Perig - 2.0;
%         elseif MissDistance > 20 && MissDistance <= 100
%             h_Perig = h_Perig - 1.0;
%         elseif MissDistance > 15 && MissDistance <= 20
%             h_Perig = h_Perig - 0.1;
%         elseif MissDistance > 5  && MissDistance <= 15
%             h_Perig = h_Perig - 0.01;
%         elseif MissDistance <= 5
%             h_Perig = h_Perig - 0.001;
%         end
%     end
% end
%

```

```

% elseif MinIndex == 2
%     if LonArrival > Lon_Target
%         if MissDistance > 100
%             h_Perig = h_Perig + 2.0;
%         elseif MissDistance > 20 && MissDistance <= 100
%             h_Perig = h_Perig + 1.0;
%         elseif MissDistance > 15 && MissDistance <= 20
%             h_Perig = h_Perig + 0.1;
%         elseif MissDistance > 5 && MissDistance <= 15
%             h_Perig = h_Perig + 0.01;
%         elseif MissDistance <= 5
%             h_Perig = h_Perig + 0.001;
%         end
%     end
%
%     elseif LonArrival < Lon_Target
%         if MissDistance > 100
%             h_Perig = h_Perig - 2.0;
%         elseif MissDistance > 20 && MissDistance <= 100
%             h_Perig = h_Perig - 1.0;
%         elseif MissDistance > 15 && MissDistance <= 20
%             h_Perig = h_Perig - 0.1;
%         elseif MissDistance > 5 && MissDistance <= 15
%             h_Perig = h_Perig - 0.01;
%         elseif MissDistance <= 5
%             h_Perig = h_Perig - 0.001;
%         end
%     end
% end
%
% % h_Perig = h_Perig
%
% WhileCount = WhileCount + 1; %Update to 'while'-loop iteration counter
%
% %Clearing of variables for targeting loop
% clear LonTGT_Crossing; clear LatTGT_Crossing; clear FlagVector_Lon;
% clear FlagVector_Lat; clear WithinIdent_Lon; clear WithinIdent_Lat;
% clear LonTGT_Jump; clear LatTGT_Jump; clear LonTGT_Interp;
% clear LatTGT_Interp;
%
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Skip Maneuver Delta-V
V_EndSkip = Skip_States(end,2); %Velocity where fpa = 0 (km/s)
dV_ReCirc = abs(V_EndSkip - V_RelProp); %Re-circularization delta-V (km/s)

%Total delta-V for skip maneuver (km/s)
dV_SkipTotal = dV_Maneuver + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Re-Defines Data to Reflect Jumps in Data between 180 and -180 deg
%Reference orbit
[Lon_RefOrb, Lat_RefOrb, LonSplit_RefOrb, LatSplit_RefOrb] = ...
    CoordinateJump(RefOrb_States);

```



```

%Maneuver orbit
[Lon_Skip, Lat_Skip, LonSplit_Skip, LatSplit_Skip] = ...
    CoordinateJump(Skip_States);

%Propagated re-circularized orbit
[Lon_PropOrb, Lat_PropOrb, LonSplit_PropOrb, LatSplit_PropOrb] = ...
    CoordinateJump(PropOrb_States);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Command Window Printing and Workspace Variable Definition
% fprintf('Minimum Miss Distance:      %f km   \n', MinDistance);
% fprintf('Time-of-Arrival:            %f hr   \n', TimeArrival);
% fprintf('Maneuver Delta-V:           %f km/s \n', dV_Maneuver);
% fprintf('Total Delta-V:              %f km/s \n', dV_SkipTotal);
%
% Trajectory_Analysis = [bank_Skip,Perigee_Altitude,h_Prop,TimeArrival, ...
%                        dV_Maneuver,dV_SkipTotal, ...
%                        -(min(PropOrb_Lat_deg)),MinDistance];
%
% Inclination_Analysis = [rad2deg(PSI_Ref),Perigee_Altitude,h_Prop, ...
%                        dV_Maneuver,dV_SkipTotal,MaxIncl];
%
% Deceleration_Analysis = [TangDecelG_Max,TangDecelG_Min, ...
%                        NormDecelG_Max,NormDecelG_Min, ...
%                        MagDecelG_Max, MagDecelG_Min];
%
% HeatFlux_Analysis    = [Qw_Max,Qs_Max,Qdot_Max,QHavey_Max,QGalman_Max];
%
% Combined_Analysis    = [Deceleration_Analysis,HeatFlux_Analysis];

% return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Commands
%Conversion of time units for plotting
[Skip_Time, time_string] = TimeUtility(Skip_t,2);
[PropOrb_Time, time_string] = TimeUtility(PropOrb_t,2);

%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,1); box on; grid off;
hold on;
h_Ref = cellfun(@plot,LonSplit_RefOrb,LatSplit_RefOrb);

set(h_Ref, 'LineStyle','-','Color','b');

xlim([-180 180]); ylim([-90 90]);
xlim([0 90]); ylim([30 70]);
% xlim([floor(Lon_Target)-30, ceil(Lon_Target)+30]);
% ylim([floor(Lat_Target)-20, ceil(Lat_Target)+20]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90]; p2_Lon = [Lon_Target, 90];

```

```

    %Target latitude, longitude lines
hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)],'r:');
hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)],'r:');

if      Map_Choice == 1
%      hold on; %Plate Carree world map projection
%      landareas = shaperead('landareas.shp','UseGeoCoords',true);
%      geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

elseif Map_Choice == 2
    hold on; %Plate Carree world map projection
    landareas = shaperead('landareas.shp','UseGeoCoords',true);
    geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);
end

%Target location
hold on; plot(Lon_Target,Lat_Target,'o','MarkerEdgeColor','r', ...
              'MarkerFaceColor','r','MarkerSize',5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,2); box on; grid off;
hold on;
h_Skip = cellfun(@plot,LonSplit_PropOrb, LatSplit_PropOrb);
hold on;
h_Ref  = cellfun(@plot,LonSplit_RefOrb,  LatSplit_RefOrb);

set(h_Skip,'LineStyle','--','Color','r');
set(h_Ref, 'LineStyle','-','Color','b');

xlim([-180 180]); ylim([-90 90]);
xlim([0 90]); ylim([30 70]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90];  p2_Lon = [Lon_Target, 90];

% %Target latitude, longitude lines
% hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)],'r:');
% hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)],'r:');

if      Map_Choice == 1
%      hold on; %Plate Carree world map projection
%      landareas = shaperead('landareas.shp','UseGeoCoords',true);
%      geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

elseif Map_Choice == 2
    hold on; %Plate Carree world map projection
    landareas = shaperead('landareas.shp','UseGeoCoords',true);
    geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);
end

```

```

%Target location
hold on; plot(Lon_Target,Lat_Target,'o','MarkerEdgeColor','r', ...
              'MarkerFaceColor','r','MarkerSize',5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) vs. Time
[PropOrb_t, time_string] = TimeUtility(PropOrb_t,2); %Time unit conversion

subplot(2,2,3); box on; grid off;
plot(PropOrb_t,PropOrb_h,'b');
xlim([0 200]); ylim([0 1000]);
xlabel(['Time, ', time_string]);
ylabel('Altitude, km');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,4); box on; grid off;
hold on;
h_Skip = cellfun(@plot,LonSplit_PropOrb, LatSplit_PropOrb);
hold on;
h_Ref = cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);

set(h_Skip,'LineStyle','--','Color','r');
set(h_Ref, 'LineStyle','-','Color','b');

xlim([-180 180]); ylim([-90 90]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90]; p2_Lon = [Lon_Target, 90];

% %Target latitude, longitude lines
% hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)],'r:');
% hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)],'r:');

if Map_Choice == 1
    % hold on; %Plate Carree world map projection
    % landareas = shaperead('landareas.shp','UseGeoCoords',true);
    % geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

elseif Map_Choice == 2
    hold on; %Plate Carree world map projection
    landareas = shaperead('landareas.shp','UseGeoCoords',true);
    geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

BankManeuvers_Function.m

```
function
[Trajectory_Analysis,MaxIncl,Deceleration_Analysis,HeatFlux_Analysis] = ...
    BankManeuvers_Function(Target_Choice,h_Perig0,PSI_Ref,bank_Skip,lon_Ref)

global MU RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape

WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Selection and Targeting Loop Initialization
LatJump_Change = 1;
LonThreshold = 35;
LatThreshold = 35;
Map_Choice = 1;

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver; Lon_Target = Lon_Denver; dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow; Lon_Target = Lon_Glasgow; dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny; Lon_Target = Lon_Grozny; dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow; Lon_Target = Lon_Moscow; dLat = 3;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti; Lon_Target = Lon_Ponti; dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong; Lon_Target = Lon_Pyong; dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk; Lon_Target = Lon_Reyk; dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran; Lon_Target = Lon_Tehran; dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo; Lon_Target = Lon_Tokyo; dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil; Lon_Target = Lon_Brasil; dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos; Lon_Target = Lon_Buenos; dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra; Lon_Target = Lon_Canberra; dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape; Lon_Target = Lon_Cape; dLat = 4;
end
```

```

h_Perig = h_Perig0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
Vehicle_Choice = 1; %TAV selection
Time_Max = 1.0; %Maximum simulation time (days)
ecc_Ref = 0; %Orbit eccentricity
h_Apog = 1000; %Apogee altitude (km)
lat_Ref = 0; %Initial geodetic latitude (deg)
fpa_Ref = 0; %Flight-path angle (deg)
bank_Ref = 0; %Reference orbit bank angle (deg)

%Converts and overwrites initial angle variables
lon_Ref = deg2rad(lon_Ref); lat_Ref = deg2rad(lat_Ref);
fpa_Ref = deg2rad(fpa_Ref); PSI_Ref = deg2rad(PSI_Ref);

%Reference orbit parameters
r_Apog = h_Apog + RE; %Apogee radial position (km)
r_Perig = h_Perig + RE; %Perigee radial position (km)
SMA_Ref = 0.5*(r_Apog + r_Perig); %Reference orbit semi-major axis (km)
SMA_Skip = 0.5*(r_Apog + r_Perig); %Skip orbit semi-major axis (km)
RefPeriod = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(Vehicle_Choice,h_Apog,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank_Ref);

%Apogee velocity for non-rotating frame (km/s)
V_Apog = sqrt((2*MU*r_Perig)/(r_Apog*(r_Apog + r_Perig)));

%Conversion of time units from days to seconds
Time_Max = Time_Max*(24)*(60)*(60);

SMA_Target0 = SMA_Ref; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel; %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
                                                1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(1,1), ...
                                                lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank_Ref);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_Apog + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
                    ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

```

```

%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
                ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
                asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
          abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
                1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(ii,1), ...
                lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank_Ref);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_Apog + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
                (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
                ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));

        %Updated heading angle (rad)
        PSI_Check0(ii+1,1) = PSI_Ref + ...
                asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
        IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
                PSI_Check0(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

```

```

V_Rel0    = V_Check0(ii);    %Velocity (km/s)
PSI_Rel0  = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t,RefOrb_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Apog,V_Rel0, ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank_Ref);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Manipulation for Reference Orbit
r_Data    = RefOrb_States(:,1); %Radial position (km)
h_Data    = r_Data - RE;        %Altitude (km)
Lon_Data  = RefOrb_States(:,3); %Longitude (rad)
Lat_Data  = RefOrb_States(:,4); %Geocentric latitude (rad)

%Transforms longitude from (0 <= lon < 360) to (-180 < lon <= 180)
Lon_Data = rem((rad2deg(Lon_Data) + 180),360) - 180;

%Converts geodetic latitude from radians to degrees
Lat_Data = rad2deg(Lat_Data);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Descent Velocity
IterMax    = 50;                %Maximum number of iterations
SMA0       = SMA_Ref;          %Initial guess for semi-major axis (km)
SMA_Target = SMA_Skip;         %Target semi-major axis (km)
V_Decrement = 1 - 0.9999;      %Decrement value for velocity (km/s)
V_Check(1,1) = .98*V_Apog;     %Initial guess for descent velocity (km/s)
% Set 'V_Check' coeff. to 0.98 for over-flights; 0.96 for max. inclination
PSI_Check(1,1) = PSI_Ref;      %Initial guess for heading angle (rad)

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[t_vec,traj_states] = Maneuver_MainFunction(Vehicle_Choice,1,2,1,1,1, ...
    0.5*RefPeriod,r_Apog,V_Check(1,1),lon_Ref, ...
    lat_Ref,fpa_Ref,PSI_Check(1,1),bank_Skip);

%Perigee radial position (km)
[r_Check(1,1),Perig_Index] = min(traj_states(:,1));

%Semi-major axis (km)
SMA_Check(1,1) = 0.5*(r_Apog + r_Check(1,1));

%Iteration error (s)
GuessError(1,1) = -((SMA_Check(1,1) - SMA0)/ ...
    ((V_Check(1,1) - V_Decrement) - V_Check(1,1)));

%Updated velocity (km/s)
V_Check(2,1) = (V_Check(1,1) - V_Decrement) - ...
    ((SMA_Target - SMA_Check(1,1))/GuessError(1,1));

%Updated heading angle (rad)
PSI_Check(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(2,1)));

```

```

%Difference between calculated and target trajectory states
IterativeDiff_SMA(1,1) = abs(SMA_Target - SMA_Check(1,1));
IterativeDiff_PSI(1,1) = abs(PSI_Check(2,1) - PSI_Check(1,1));

IterCount = 1; %Initialization of iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target - SMA_Check(ii-1,1)) > 1E-6 && ...
        abs(PSI_Check(ii,1) - PSI_Check(ii-1,1)) > 1E-8

        %Trajectory simulation [0:t:HalfPeriod]
        [t_vec,traj_states] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,2,1,1,1,0.5*RefPeriod,r_Apog, ...
            V_Check(ii,1),lon_Ref,lat_Ref,fpa_Ref, ...
            PSI_Check(ii,1),bank_Skip);

        %Current iteration perigee radial position (km)
        [r_Check(ii,1),Perig_Index] = min(traj_states(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check(ii,1) = 0.5*(r_Apog + r_Check(ii,1));

        %Iteration error (sec)
        GuessError(ii,1) = -((SMA_Check(ii,1) - SMA_Check(ii-1,1))/ ...
            (V_Check(ii,1) - V_Check(ii-1,1)));

        %Updated velocity (km/s)
        V_Check(ii+1,1) = V_Check(ii,1) - ...
            ((SMA_Target - SMA_Check(ii,1))/GuessError(ii,1));

        %Updated heading angle (rad)
        PSI_Check(ii+1,1) = PSI_Ref + ...
            asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA(ii,1) = abs(SMA_Target - SMA_Check(ii,1));
        IterativeDiff_PSI(ii,1) = abs(PSI_Check(ii,1) - PSI_Check(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

V_Maneuver = V_Check(ii,1); %Descent velocity for target SMA
dV_Maneuver = abs(V_Maneuver - V_Rel); %Maneuver delta-V (km/s)
PSI_Rel = rad2deg(PSI_Check(ii,1)); %Relative heading angle (deg)

```



```

%Trajectory simulation for skip maneuver
[Skip_t,Skip_States] = SingleSkip_Maneuver(Vehicle_Choice,1,2,1,1,1, ...
    RefPeriod,r_Apog,V_Maneuver,lon_Ref, ...
    lat_Ref,fpa_Ref,deg2rad(PSI_Rel),bank_Skip);

Perigee_Altitude = min(Skip_States(:,1)) - RE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Propagation of Re-Circularized Orbit
Time_Max = 80000; %Maximum simulation time (s)
ecc = 0; %Orbit eccentricity
r_Prop = Skip_States(end,1); %Orbit radial position (km)
h_Prop = r_Prop - RE; %Orbit altitude (km)
lon_Prop = Skip_States(end,3); %Initial longitude (rad)
lat_Prop = Skip_States(end,4); %Initial geodetic latitude (rad)
fpa_Prop = 0; %Flight-path angle (rad)
PSI_Prop = -(min(Skip_States(:,4))); %Heading angle (rad)
bank_Prop = bank_Skip; %Bank angle (deg)

%Re-circularized orbit parameters
SMA_Prop = 0.5*(r_Prop + r_Prop); %Semi-major axis (km)
Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_RelProp,PSI_RelProp] = RelativeStates(Vehicle_Choice,h_Prop,lat_Prop, ...
    fpa_Prop,PSI_Prop,bank_Prop);

SMA_TargetProp = SMA_Prop;
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckP(1,1) = V_RelProp; %Initial guess for velocity (km/s)
PSI_CheckP(1,1) = PSI_RelProp; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP,Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
    V_CheckP(1,1),lon_Prop,lat_Prop, ...
    fpa_Prop,PSI_CheckP(1,1),bank_Prop);

[r_CheckP(1,1),ApogFlag] = min(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckP(1,1) = 0.5*(r_Prop + r_CheckP(1,1));

%Iteration error (s)
GuessErrorP(1,1) = -((SMA_CheckP(1,1) - SMA_Prop)/ ...
    ((V_CheckP(1,1) - V_Decrement) - V_CheckP(1,1)));

%Updated velocity (km/s)
V_CheckP(2,1) = (V_CheckP(1,1) - V_Decrement) - ...
    ((SMA_TargetProp - SMA_CheckP(1,1))/GuessErrorP(1,1));

```

```

%Updated heading angle (rad)
PSI_CheckP(2,1) = PSI_Prop + ...
    asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckP(1,1));
IterativeDiff_PSI_Prop(1,1) = abs(PSI_CheckP(2,1) - PSI_CheckP(1,1));
IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckP(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckP(ii,1) - PSI_CheckP(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP,Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckP(ii,1), ...
            lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii,1),bank_Prop);

        [r_CheckP(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

        %Current iteration semi-major axis (km)
        SMA_CheckP(ii,1) = 0.5*(r_Prop + r_CheckP(ii,1));

        %Iteration error (sec)
        GuessErrorP(ii,1) = -((SMA_CheckP(ii,1) - SMA_CheckP(ii-1,1))/ ...
            (V_CheckP(ii,1) - V_CheckP(ii-1,1)));

        %Updated velocity (km/s)
        V_CheckP(ii+1,1) = V_CheckP(ii,1) - ...
            ((SMA_TargetProp - SMA_CheckP(ii,1))/GuessErrorP(ii,1));

        %Updated heading angle (rad)
        PSI_CheckP(ii+1,1) = PSI_Prop + ...
            asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp - ...
            SMA_CheckP(ii,1));
        IterativeDiff_PSI_Prop(ii,1) = abs(PSI_CheckP(ii,1) - ...
            PSI_CheckP(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t,Orbit_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Prop,V_CheckP(ii), ...
    lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii),bank_Prop);

```

```

%Re-defined propagated orbit states
PropOrb_t      = [Skip_t ; Skip_t(end) + Orbit_t(2:end)];
PropOrb_States = [Skip_States(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h      = PropOrb_States(:,1) - RE;      %Altitude (km)
PropOrb_V      = PropOrb_States(:,2);          %Velocity (km/s)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
                      + 180),360) - 180;        %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Propagated Trajectory Crossings of Target Coordinates
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lon_deg)
    mm = mm + 1;
    if abs(PropOrb_Lon_deg(ii) - Lon_Target) < LonThreshold
        LonTGT_Crossing(mm,1) = PropOrb_t(ii);
        LonTGT_Crossing(mm,2) = PropOrb_h(ii);
        LonTGT_Crossing(mm,3) = PropOrb_Lat_deg(ii);
        LonTGT_Crossing(mm,4) = PropOrb_Lon_deg(ii);
    else
        LonTGT_Crossing(mm,1:4) = 0;
    end
end

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lat_deg)
    mm = mm + 1;
    if abs(PropOrb_Lat_deg(ii) - Lat_Target) < LatThreshold
        LatTGT_Crossing(mm,1) = PropOrb_t(ii);
        LatTGT_Crossing(mm,2) = PropOrb_h(ii);
        LatTGT_Crossing(mm,3) = PropOrb_Lon_deg(ii);
        LatTGT_Crossing(mm,4) = PropOrb_Lat_deg(ii);
    else
        LatTGT_Crossing(mm,1:4) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Crossings
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(LonTGT_Crossing)
    if LonTGT_Crossing(ii) ~= 0
        mm = mm + 1;
        FlagVector_Lon(mm,1) = ii;
        WithinIdent_Lon(mm,:) = LonTGT_Crossing(ii,:);
    end
end
FlagVector_Lon = [FlagVector_Lon;0];

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero

```

```

for ii = 1:length(LatTGT_Crossing)
    if LatTGT_Crossing(ii) ~= 0
        mm = mm + 1;
        FlagVector_Lat(mm,1) = ii;
        WithinIdent_Lat(mm,:) = LatTGT_Crossing(ii,:);
    end
end
FlagVector_Lat = [FlagVector_Lat;0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Crossings
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Lon)-1
    if abs((FlagVector_Lon(ii+1) - FlagVector_Lon(ii))) > 1
        mm = mm + 1;
        LonTGT_Jump(mm,1) = ii;
    end
end
LonTGT_Jump = [0;LonTGT_Jump];

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Lat)-1
    if abs((FlagVector_Lat(ii+1) - FlagVector_Lat(ii))) > 1
        mm = mm + 1;
        LatTGT_Jump(mm,1) = ii;
    end
end

if LatJump_Change == 1 %Appropriate for 'mid-' to 'high-' latitudes
    LatTGT_Jump = [0; LatTGT_Jump];
elseif LatJump_Change == 2 %Appropriate for 'low-' latitudes
    LatTGT_Jump = [LatTGT_Jump];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Crossing Trajectories
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(LonTGT_Jump)
    mm = mm + 1;
    LonTGT_Interp(mm,:) = ...
        interp1(LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
            FlagVector_Lon(LonTGT_Jump(ii)),4), ...
            LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
            FlagVector_Lon(LonTGT_Jump(ii)),1:3), ...
            Lon_Target, 'spline'); %Cubic spline interpolation
end

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(LatTGT_Jump)
    mm = mm + 1;
    LatTGT_Interp(mm,:) = ...

```

```

        interp1(LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
                FlagVector_Lat(LatTGT_Jump(ii)),4), ...
                LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
                FlagVector_Lat(LatTGT_Jump(ii)),1:3), ...
                Lat_Target, 'spline'); %Cubic spline interpolation
end

%Removal of negative perturbed periods
LatTGT_Interp(any(LatTGT_Interp(:,1)<0,2),:) = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Minimum Target Miss Distance
%Target miss distance for both spherical and oblate planetary models
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lon_deg)
    mm = mm + 1;
    SphereDist_Lon = CoordDist(Lon_Target,Lon_Target, ...
                                Lat_Target,LonTGT_Interp(:,3),1);
end

%Longitudinal target miss distance (km)
[MinDistance_Lon,MinFlag_Lon] = min(SphereDist_Lon(:,1));

mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lat_deg)
    mm = mm + 1;
    SphereDist_Lat = CoordDist(Lon_Target,LatTGT_Interp(:,3), ...
                                Lat_Target,Lat_Target,1);
end

%Latitudinal target miss distance (km)
[MinDistance_Lat,MinFlag_Lat] = min(SphereDist_Lat(:,1));

MinDist_Vec = [MinDistance_Lon, MinDistance_Lat];
MinFlag_Vec = [MinFlag_Lon,      MinFlag_Lat];

[MinDistance, MinIndex] = min(MinDist_Vec);
MinFlag       = MinFlag_Vec(MinIndex);
MissDistance = MinDistance;

if      MinIndex == 1
    MinInterp = LonTGT_Interp;
elseif MinIndex == 2
    MinInterp = LatTGT_Interp;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Over-Flight Parameters
%Time-of-arrival at target (hr)
TimeArrival = (MinInterp(MinFlag,1))*(1/60)*(1/60);

```

```

%Altitude-of-arrival at target (km)
AltArrival = MinInterp(MinFlag,2);

%Payload imager field-of-view (FOV) and resolution during over-flight
%Visible spectrum imager
[FOV_m2_Vis,FOV_km2_Vis,Resolution_Vis] = ...
    PayloadImager(AltArrival*(1.0E3),1.15,2.70,1.0E-6);

%Latitude-of-arrival at target (deg)
LatArrival = LonTGT_Interp(MinFlag_Lon,3);

%Longitude-of-arrival at target (deg)
LonArrival = LatTGT_Interp(MinFlag_Lat,3);

%Maximum inclination (deg)
MaxIncl = max(PropOrb_Lat_deg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
[Vehicle] = VehicleSpecs(Vehicle_Choice);

mass = Vehicle.mass; %Mass (kg)
S_m2 = Vehicle.S_m2; %Planform area (m^2)
S = S_m2/(1000^2); %Planform area (km^2)
Cd = Vehicle.Cd; %Drag coefficient
Cl = Vehicle.Cl; %Lift coefficient

[decel] = EntryDecel(1,mass,S,Cd,Cl,
    PropOrb_States(:,1),PropOrb_States(:,2), ...
    PropOrb_States(:,4),PropOrb_States(:,5));

%Tangential deceleration (g's)
TangDecelG_Max = max(decel.TangG); %Maximum value
TangDecelG_Min = min(decel.TangG); %Minimum value

%Normal deceleration (g's)
NormDecelG_Max = max(decel.NormalG); %Maximum value
NormDecelG_Min = min(decel.NormalG); %Minimum value

%Deceleration magnitude (g's)
MagDecelG_Max = max(decel.Gs); %Maximum value
MagDecelG_Min = min(decel.Gs); %Minimum value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Heat Flux
%Atmospheric density (kg/km^3)
[PropOrb_Rho] = AtmosModel_PostAnalysis(PropOrb_h,2);

%Maximum velocity (km/s) -- Occurs at Perigee
VMax = max(PropOrb_V);
Emissivity = 0.8; %Emissivity
Tw_F = 0; %Wall temperature (deg F)
TMaxF = 1800; %Free-stream temperature (deg F)

```

```

%Heat transfer models
[HeatModel,Eta,T_KE] = HeatFluxModel(Vehicle_Choice,PropOrb_V, ...
                                     PropOrb_Rho,Emissivity,Tw_F,TMaxF);

%Average wall heat flux (non-dimensional)
Qw          = HeatModel.Qw;
Qw_Max      = max(Qw);    %Maximum value

%Average stagnation heat flux (non-dimensional)
Qs          = HeatModel.Qs;
Qs_Max      = max(Qs);    %Maximum value

%Stagnation heat flux (kW/m^2); Source: Rao (2002)
Qdot        = HeatModel.Qdot;
Qdot_Max    = max(Qdot); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Havey (1982)
QHavey      = HeatModel.QHavey;
QHavey_Max  = max(QHavey);

%Stagnation heat flux (kW/m^2); Source: Galman (1961)
QGalman     = HeatModel.QGalman;
QGalman_Max = max(QGalman);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Skip Maneuver Delta-V
V_EndSkip = Skip_States(end,2); %Velocity where fpa = 0 (km/s)
dV_ReCirc = abs(V_EndSkip - V_RelProp); %Re-circularization delta-V (km/s)

%Total delta-V for skip maneuver (km/s)
dV_SkipTotal = dV_Maneuver + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Workspace Variable Definition
Trajectory_Analysis = [bank_Skip,Perigee_Altitude,h_Prop,TimeArrival, ...
                      dV_Maneuver,dV_SkipTotal, ...
                      -(min(PropOrb_Lat_deg)),MinDistance];

Inclination_Analysis = [rad2deg(PSI_Ref),Perigee_Altitude,h_Prop, ...
                      dV_Maneuver,dV_SkipTotal,MaxIncl];

Deceleration_Analysis = [TangDecelG_Max,TangDecelG_Min, ...
                        NormDecelG_Max,NormDecelG_Min, ...
                        MagDecelG_Max, MagDecelG_Min];

HeatFlux_Analysis = [Qw_Max,Qs_Max,Qdot_Max,QHavey_Max,QGalman_Max];

Combined_Analysis = [HeatFlux_Analysis,Deceleration_Analysis];

```

BankManeuvers_fxnDOE.m

```
function
[Trajectory_Analysis,MaxIncl,Deceleration_Analysis,HeatFlux_Analysis] = ...
    BankManeuvers_fxnDOE(Vehicle_Choice,Target_Choice,lon_Ref, ...
        PSI_Ref,bank_Skip,Factor_mass,Factor_S,Factor_Cd, ...
        Factor_Cl,Factor_Perig,Factor_InitAlt)

global MU RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape

WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Selection and Targeting Loop Initialization
LatJump_Change = 1; LonThreshold = 35; LatThreshold = 35;

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver; Lon_Target = Lon_Denver; dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow; Lon_Target = Lon_Glasgow; dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny; Lon_Target = Lon_Grozny; dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow; Lon_Target = Lon_Moscow; dLat = 3;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti; Lon_Target = Lon_Ponti; dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong; Lon_Target = Lon_Pyong; dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk; Lon_Target = Lon_Reyk; dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran; Lon_Target = Lon_Tehran; dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo; Lon_Target = Lon_Tokyo; dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil; Lon_Target = Lon_Brasil; dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos; Lon_Target = Lon_Buenos; dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra; Lon_Target = Lon_Canberra; dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape; Lon_Target = Lon_Cape; dLat = 4;
end
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 99    %VEHICLE SELECTION OVERRIDE
    mass = Factor_mass;    %Mass (kg)
    S_m2 = Factor_S;       %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Factor_Cd;     %Drag coefficient
    Cl    = Factor_Cl;     %Lift coefficient
else
    [Vehicle] = VehicleSpecs(Vehicle_Choice);
    mass = Vehicle.mass;    %Mass (kg)
    S_m2 = Vehicle.S_m2;    %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Vehicle.Cd;     %Drag coefficient
    Cl    = Vehicle.Cl;     %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
Time_Max = 1.0;            %Maximum simulation time (days)
ecc_Ref   = 0;             %Orbit eccentricity
h_Perig   = Factor_Perig;  %Perigee altitude (km)
h_Apog    = Factor_InitAlt; %Apogee altitude (km)
lat_Ref   = 0;            %Initial geodetic latitude (deg)
fpa_Ref   = 0;            %Flight-path angle (deg)
bank_Ref  = 0;            %Reference orbit bank angle (deg)

%Converts and overwrites initial angle variables
lon_Ref = deg2rad(lon_Ref); lat_Ref = deg2rad(lat_Ref);
fpa_Ref = deg2rad(fpa_Ref); PSI_Ref = deg2rad(PSI_Ref);

%Reference orbit parameters
r_Apog   = h_Apog + RE;    %Apogee radial position (km)
r_Perig  = h_Perig + RE;   %Perigee radial position (km)
SMA_Ref  = 0.5*(r_Apog + r_Perig); %Reference orbit semi-major axis (km)
SMA_Skip = 0.5*(r_Apog + r_Perig); %Skip orbit semi-major axis (km)
RefPeriod = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_Apog,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank_Ref);

%Apogee velocity for non-rotating frame (km/s)
V_Apog = sqrt((2*MU*r_Perig)/(r_Apog*(r_Apog + r_Perig)));

%Conversion of time units from days to seconds
Time_Max = Time_Max*(24)*(60)*(60);

SMA_Target0 = SMA_Ref;    %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel;    %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref; %Initial guess for heading angle (rad)
IterMax       = 50;       %Maximum number of iterations

```

```

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0,Traj_States0] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
    1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(1,1), ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1), ...
    bank_Ref,Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_Apog + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
    ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));
%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
    ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0,Traj_States0] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
            1,1,1,1,1,0.5*RefPeriod,r_Apog,V_Check0(ii,1), ...
            lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1), ...
            bank_Ref,Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_Apog + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
            (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
            ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));
    end
end

```

```

    %Updated heading angle (rad)
    PSI_Check0(ii+1,1) = PSI_Ref + ...
        asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

    %Difference between calculated and target trajectory states
    IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
    IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
        PSI_Check0(ii-1,1));

    ii = ii + 1; %Update to row-index counter
    IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t,RefOrb_States] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Apog,V_Rel0,lon_Ref, ...
    lat_Ref,fpa_Ref,PSI_Rel0,bank_Ref, ...
    Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Manipulation for Reference Orbit
r_Data = RefOrb_States(:,1); %Radial position (km)
h_Data = r_Data - RE; %Altitude (km)
Lon_Data = RefOrb_States(:,3); %Longitude (rad)
Lat_Data = RefOrb_States(:,4); %Geocentric latitude (rad)

%Transforms longitude from (0 <= lon < 360) to (-180 < lon <= 180)
Lon_Data = rem((rad2deg(Lon_Data) + 180),360) - 180;

%Converts geodetic latitude from radians to degrees
Lat_Data = rad2deg(Lat_Data);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Descent Velocity
IterMax = 20; %Maximum number of iterations
SMA0 = SMA_Ref; %Initial guess for semi-major axis (km)
SMA_Target = SMA_Skip; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check(1,1) = .95*V_Apog; %Initial guess for descent velocity (km/s)
%Set 'V_Check' coeff. to 0.98 for over-flights; 0.96 for max. inclination
PSI_Check(1,1) = PSI_Ref; %Initial guess for heading angle (rad)

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[t_vec,traj_states] = Maneuver_MainFunctionDOE(Vehicle_Choice,1,2,1,1,1, ...
    0.5*RefPeriod,r_Apog,V_Check(1,1),lon_Ref, ...
    lat_Ref,fpa_Ref,PSI_Check(1,1),bank_Skip, ...
    Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

```

```

%Perigee radial position (km)
[r_Check(1,1),Perig_Index] = min(traj_states(:,1));

%Semi-major axis (km)
SMA_Check(1,1) = 0.5*(r_Apog + r_Check(1,1));

%Iteration error (s)
GuessError(1,1) = -((SMA_Check(1,1) - SMA0)/ ...
    ((V_Check(1,1) - V_Decrement) - V_Check(1,1)));

%Updated velocity (km/s)
V_Check(2,1) = (V_Check(1,1) - V_Decrement) - ...
    ((SMA_Target - SMA_Check(1,1))/GuessError(1,1));

%Updated heading angle (rad)
PSI_Check(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA(1,1) = abs(SMA_Target - SMA_Check(1,1));
IterativeDiff_PSI(1,1) = abs(PSI_Check(2,1) - PSI_Check(1,1));

IterCount = 1; %Initialization of iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target - SMA_Check(ii-1,1)) > 1E-6 && ...
        abs(PSI_Check(ii,1) - PSI_Check(ii-1,1)) > 1E-8 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [t_vec,traj_states] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
            1,2,1,1,1,0.5*RefPeriod,r_Apog, ...
            V_Check(ii,1),lon_Ref,lat_Ref,fpa_Ref, ...
            PSI_Check(ii,1),bank_Skip,Factor_mass, ...
            Factor_S,Factor_Cd,Factor_Cl);

        if traj_states(end,2) < 1
            traj_states(:,2) = 0;
            %Limits time vector length to length of traj. parameter matrix
            t_vec = zeros(length(traj_states(:,1)),1);
            break
        end

        %Current iteration perigee radial position (km)
        [r_Check(ii,1),Perig_Index] = min(traj_states(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check(ii,1) = 0.5*(r_Apog + r_Check(ii,1));

        %Iteration error (sec)
        GuessError(ii,1) = -((SMA_Check(ii,1) - SMA_Check(ii-1,1))/ ...
            (V_Check(ii,1) - V_Check(ii-1,1)));

```

```

    %Updated velocity (km/s)
    V_Check(ii+1,1) = V_Check(ii,1) - ...
        ((SMA_Target - SMA_Check(ii,1))/GuessError(ii,1));

    %Updated heading angle (rad)
    PSI_Check(ii+1,1) = PSI_Ref + ...
        asin((2*pi*(r_Apog)*sin(PSI_Ref))/(86400*V_Check(ii+1,1)));

    %Difference between calculated and target trajectory states
    IterativeDiff_SMA(ii,1) = abs(SMA_Target - SMA_Check(ii,1));
    IterativeDiff_PSI(ii,1) = abs(PSI_Check(ii,1) - PSI_Check(ii-1,1));

    ii = ii + 1; %Update to row-index counter
    IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Maneuver = V_Check(ii,1); %Descent velocity for target SMA
dV_Maneuver = abs(V_Maneuver - V_Rel); %Maneuver delta-V (km/s)
PSI_Rel = rad2deg(PSI_Check(ii,1)); %Relative heading angle (deg)

if traj_states(end,1) == 0
    Skip_t = 0;
    Skip_States = zeros(1,8);
    Perigee_Altitude = 0;

else
    %Trajectory simulation for skip maneuver
    [Skip_t,Skip_States] = SingleSkip_ManeuverDOE(Vehicle_Choice,1,2,1,1,1, ...
        RefPeriod,r_Apog,V_Maneuver,lon_Ref, ...
        lat_Ref,fpa_Ref,deg2rad(PSI_Rel),bank_Skip, ...
        Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

    Perigee_Altitude = min(Skip_States(:,1)) - RE;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Perigee_Altitude > 50
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% Propagation of Re-Circularized Orbit
    Time_Max = 7200; %Maximum simulation time (s)
    ecc = 0; %Orbit eccentricity
    r_Prop = Skip_States(end,1); %Orbit radial position (km)
    h_Prop = r_Prop - RE; %Orbit altitude (km)
    lon_Prop = Skip_States(end,3); %Initial longitude (rad)
    lat_Prop = Skip_States(end,4); %Initial geodetic latitude (rad)
    fpa_Prop = 0; %Flight-path angle (rad)
    PSI_Prop = -(min(Skip_States(:,4))); %Heading angle (rad)
    bank_Prop = bank_Skip; %Bank angle (deg)

    %Re-circularized orbit parameters
    SMA_Prop = 0.5*(r_Prop + r_Prop); %Semi-major axis (km)
    Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU); %Orbit period (sec)

```

```

%Velocity relative to rotating frame (rotating planet)
[V_RelProp,PSI_RelProp] = RelativeStates(mass,S,Cd,Cl,h_Prop,lat_Prop, ...
                                         fpa_Prop,PSI_Prop,bank_Prop);

SMA_TargetProp = SMA_Prop;
V_Decrement    = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckP(1,1)   = V_RelProp;  %Initial guess for velocity (km/s)
PSI_CheckP(1,1) = PSI_RelProp; %Initial guess for heading angle (rad)
IterMax        = 50;          %Maximum number of iterations
%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP,Traj_StatesP] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
    1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
    V_CheckP(1,1),lon_Prop,lat_Prop, ...
    fpa_Prop,PSI_CheckP(1,1),bank_Prop, ...
    Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

[r_CheckP(1,1),ApogFlag] = min(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckP(1,1) = 0.5*(r_Prop + r_CheckP(1,1));

%Iteration error (s)
GuessErrorP(1,1) = -((SMA_CheckP(1,1) - SMA_Prop)/ ...
    ((V_CheckP(1,1) - V_Decrement) - V_CheckP(1,1)));

%Updated velocity (km/s)
V_CheckP(2,1) = (V_CheckP(1,1) - V_Decrement) - ...
    ((SMA_TargetProp - SMA_CheckP(1,1))/GuessErrorP(1,1));

%Updated heading angle (rad)
PSI_CheckP(2,1) = PSI_Prop + ...
    asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckP(1,1));
IterativeDiff_PSI_Prop(1,1) = abs(PSI_CheckP(2,1) - PSI_CheckP(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckP(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckP(ii,1) - PSI_CheckP(ii-1,1)) > 1E-10 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP,Traj_StatesP] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
            V_CheckP(ii,1),lon_Prop,lat_Prop, ...
            fpa_Prop,PSI_CheckP(ii,1),bank_Prop, ...
            Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

```

```

[r_CheckP(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

%Current iteration semi-major axis (km)
SMA_CheckP(ii,1) = 0.5*(r_Prop + r_CheckP(ii,1));

%Iteration error (sec)
GuessErrorP(ii,1) = -((SMA_CheckP(ii,1) - SMA_CheckP(ii-1,1))/ ...
    (V_CheckP(ii,1) - V_CheckP(ii-1,1)));

%Updated velocity (km/s)
V_CheckP(ii+1,1) = V_CheckP(ii,1) - ...
    ((SMA_TargetProp - SMA_CheckP(ii,1))/GuessErrorP(ii,1));

%Updated heading angle (rad)
PSI_CheckP(ii+1,1) = PSI_Prop + ...
    asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(ii+1,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp - ...
    SMA_CheckP(ii,1));
IterativeDiff_PSI_Prop(ii,1) = abs(PSI_CheckP(ii,1) - ...
    PSI_CheckP(ii-1,1));

ii = ii + 1; %Update to row-index counter
IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t,Orbit_States] = Maneuver_MainFunctionDOE(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Prop,V_CheckP(ii), ...
    lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii), ...
    bank_Prop,Factor_mass,Factor_S,Factor_Cd,Factor_Cl);

%Re-defined propagated orbit states
PropOrb_t = [Skip_t ; Skip_t(end) + Orbit_t(2:end)];
PropOrb_States = [Skip_States(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h = PropOrb_States(:,1) - RE; %Altitude (km)
PropOrb_V = PropOrb_States(:,2); %Velocity (km/s)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
    + 180),360) - 180; %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

%Maximum inclination (deg)
[MaxIncl,MaxIndex] = max(PropOrb_Lat_deg);

%Time-of-flight to reach maximum inclination (hr)
TimeMaxIncl = PropOrb_t(MaxIndex)*(1/60)*(1/60);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
%Vehicle model
if Vehicle_Choice < 99
    [Vehicle] = VehicleSpecs(Vehicle_Choice);

    mass = Vehicle.mass; %Mass (kg)
    S_m2 = Vehicle.S_m2; %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Vehicle.Cd; %Drag coefficient
    Cl    = Vehicle.Cl; %Lift coefficient
elseif Vehicle_Choice == 99
    mass = Factor_mass; %Mass (kg)
    S_m2 = Factor_S; %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Factor_Cd; %Drag coefficient
    Cl    = Factor_Cl; %Lift coefficient
end

[decel] = EntryDecel(1,mass,S,Cd,Cl,
                    PropOrb_States(:,1),PropOrb_States(:,2), ...
                    PropOrb_States(:,4),PropOrb_States(:,5));

%Tangential deceleration (g's)
TangDecelG_Max = max(decel.TangG); %Maximum value
TangDecelG_Min = min(decel.TangG); %Minimum value
%Normal deceleration (g's)
NormDecelG_Max = max(decel.NormalG); %Maximum value
NormDecelG_Min = min(decel.NormalG); %Minimum value
%Deceleration magnitude (g's)
MagDecelG_Max = max(decel.Gs); %Maximum value
MagDecelG_Min = min(decel.Gs); %Minimum value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Heat Flux
%Atmospheric density (kg/km^3)
[PropOrb_Rho] = AtmosModel_PostAnalysis(PropOrb_h,2);

%Maximum velocity (km/s) -- Occurs at Perigee
VMax = max(PropOrb_V);
Emissivity = 0.8; %Emissivity
Tw_F = 0; %Wall temperature (deg F)
TMaxF = 1800; %Free-stream temperature (deg F)

%Heat transfer models
[HeatModel,Eta,T_KE] = HeatFluxModel(PropOrb_V,PropOrb_Rho,Emissivity, ...
                                     Tw_F,TMaxF,mass,S,Cd,Cl);

%Average wall heat flux (non-dimensional)
Qw = HeatModel.Qw;
Qw_Max = max(Qw); %Maximum value
%Average stagnation heat flux (non-dimensional)
Qs = HeatModel.Qs;
Qs_Max = max(Qs); %Maximum value

```



```

%Stagnation heat flux (kW/m^2); Source: Rao (2002)
Qdot      = HeatModel.Qdot;
Qdot_Max  = max(Qdot); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Havey (1982)
QHavey    = HeatModel.QHavey;
QHavey_Max = max(QHavey);

%Stagnation heat flux (kW/m^2); Source: Galman (1961)
QGalman   = HeatModel.QGalman;
QGalman_Max = max(QGalman);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Skip Maneuver Delta-V
V_EndSkip = Skip_States(end,2); %Velocity where fpa = 0 (km/s)
dV_ReCirc = abs(V_EndSkip - V_RelProp); %Re-circularization delta-V (km/s)

%Total delta-V for skip maneuver (km/s)
dV_SkipTotal = dV_Maneuver + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Workspace Variable Definition
%Dimension (7x1)
Trajectory_Analysis = [bank_Skip,Perigee_Altitude,h_Prop,PropOrb_h(end), ...
    TimeMaxIncl,dV_Maneuver,dV_SkipTotal];

%Dimension (5x1)
Inclination_Analysis = [rad2deg(PSI_Ref),Perigee_Altitude,h_Prop, ...
    dV_Maneuver,dV_SkipTotal,MaxIncl];

%Dimension (6x1)
Deceleration_Analysis = [TangDecelG_Max,TangDecelG_Min, ...
    NormDecelG_Max,NormDecelG_Min, ...
    MagDecelG_Max, MagDecelG_Min];

%Dimension (5x1)
HeatFlux_Analysis = [Qw_Max,Qs_Max,Qdot_Max,QHavey_Max,QGalman_Max];

%Dimension (11x1)
Combined_Analysis = [Deceleration_Analysis,HeatFlux_Analysis];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif Perigee_Altitude <= 50
%Dimension (7x1)
Trajectory_Analysis = zeros(7,1);
%Dimension (5x1)
Inclination_Analysis = zeros(5,1);
%Dimension (1x1)
MaxIncl = zeros(1,1);
%Dimension (6x1)
Deceleration_Analysis = zeros(6,1);
%Dimension (5x1)
HeatFlux_Analysis = zeros(5,1);
%Dimension (11x1)
Combined_Analysis = zeros(11,1);
end

```

BankManeuvers_fxnDOE_Hohmann.m

```
function
[Skip_t, Skip_States, Trajectory_States, RefOrb_States, Trajectory_Analysis] = ...
    BankManeuvers_MultiAOT(Vehicle_Choice, Time_Max, h_init, ...
        PSI_Ref, fpa_Descent, dV_Boost, bank_Skip)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 9      %VEHICLE SELECTION OVERRIDE
    mass = 2000;            %Mass (kg)
    S_m2 = 18.5;            %Planform area (m^2)
    S     = S_m2/(1000^2);  %Planform area (km^2)
    Cd    = 0.5;            %Drag coefficient
    Cl    = 3.0;            %Lift coefficient
else
    [Vehicle] = VehicleSpecs(Vehicle_Choice);
    mass = Vehicle.mass;    %Mass (kg)
    S_m2 = Vehicle.S_m2;    %Planform area (m^2)
    S     = S_m2/(1000^2);  %Planform area (km^2)
    Cd    = Vehicle.Cd;     %Drag coefficient
    Cl    = Vehicle.Cl;     %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
h_atm    = 120; %Altitude of upper limit of sensible atmosphere (km)
lon_Ref   = 0;  %Initial longitude (deg)
lat_Ref   = 0;  %Initial geodetic latitude (deg)
fpa_Ref   = 0;  %Flight-path angle (deg)
bank_Ref  = 0;  %Reference orbit bank angle (deg)

%Converts and overwrites initial angle variables
lon_Ref   = deg2rad(lon_Ref);    lat_Ref = deg2rad(lat_Ref);
fpa_Ref   = deg2rad(fpa_Ref);    PSI_Ref = deg2rad(PSI_Ref);
fpa_Descent = deg2rad(fpa_Descent);

%Reference orbit parameters
r_init    = h_init + RE;          %Initial radial position (km)
SMA_Ref   = 0.5*(r_init + r_init); %Reference orbit semi-major axis (km)
RefPeriod = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel, PSI_Rel] = RelativeStates(mass, S, Cd, Cl, h_init, lat_Ref, ...
    fpa_Ref, PSI_Ref, bank_Ref);

%Conversion of time units from minutes to seconds
Time_Max = Time_Max*(60);
SMA_Target0 = SMA_Ref; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
```

```

V_Check0(1,1)    = V_Rel;           %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref;         %Initial guess for heading angle (rad)
IterMax         = 50;              %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
        1,1,1,1,1,0.5*RefPeriod,r_init,V_Check0(1,1), ...
        lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank_Ref);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_init + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
        ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

%Updated velocity (km/s)
V_Check0(2,1)    = (V_Check0(1,1) - V_Decrement) - ...
        ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
        asin((2*pi*(r_init)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
                1,1,1,1,1,0.5*RefPeriod,r_init,V_Check0(ii,1), ...
                lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank_Ref);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_init + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
                (V_Check0(ii,1) - V_Check0(ii-1,1)));

```

```

%Updated velocity (km/s)
V_Check0(ii+1,1) = V_Check0(ii,1) - ...
    ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));

%Updated heading angle (rad)
PSI_Check0(ii+1,1) = PSI_Ref + ...
    asin((2*pi*(r_init)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
    PSI_Check0(ii-1,1));

ii = ii + 1; %Update to row-index counter
IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t,RefOrb_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_init,V_Rel0, ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank_Ref);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Skip Entry Manuever
%Relative states descent-boost maneuver
[V_RelBoost,fpa_RelBoost,PSI_RelBoost] = ...
    RelativeStates_Entry(h_init,dV_Boost,lon_Ref, ...
    lat_Ref,fpa_Descent,PSI_Ref);

%Trajectory simulation for skip maneuvers
[Skip_t,Skip_States] = Maneuver_MainFunction(Vehicle_Choice,1,1,1,1,1, ...
    Time_Max,r_init,V_RelBoost,lon_Ref, ...
    lat_Ref,fpa_RelBoost,PSI_RelBoost,bank_Skip);

%Skip entry trajectory states
SkipTraj_h = (Skip_States(:,1)) - RE; %Altitude (km)
SkipTraj_V = Skip_States(:,2); %Velocity (km/s)

%Re-Circularized velocity relative to rotating frame (rotating planet)
[V_RelProp,PSI_RelProp] = ...
    RelativeStates(mass,S,Cd,Cl,SkipTraj_h(end),Skip_States(end,3), ...
    Skip_States(end,5),Skip_States(end,6),bank_Skip);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
[decel] = EntryDecel(1,mass,S,Cd,Cl,Skip_States(:,1),Skip_States(:,2), ...
    Skip_States(:,4),Skip_States(:,5));

```

```

%Tangential deceleration (g's)
TangDecelG_Max = max(decel.TangG); %Maximum value
TangDecelG_Min = min(decel.TangG); %Minimum value

%Normal deceleration (g's)
NormDecelG_Max = max(decel.NormalG); %Maximum value
NormDecelG_Min = min(decel.NormalG); %Minimum value

%Deceleration magnitude (g's)
MagDecelG_Max = max(decel.Gs); %Maximum value
MagDecelG_Min = min(decel.Gs); %Minimum value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Heat Flux
%Atmospheric density (kg/km^3)
[SkipTraj_Rho] = AtmosModel_PostAnalysis(SkipTraj_h,2);

%Maximum velocity (km/s) -- Occurs at Perigee
VMax = max(SkipTraj_V);
Emissivity = 0.8; %Emissivity
Tw_F = 0; %Wall temperature (deg F)
TMaxF = 1800; %Free-stream temperature (deg F)

%Heat transfer models
[HeatModel,Eta,T_KE] = HeatFluxModel(SkipTraj_V,SkipTraj_Rho,Emissivity, ...
                                     Tw_F,TMaxF,mass,S,Cd,Cl);

%Average wall heat flux (non-dimensional)
Qw = HeatModel.Qw;
Qw_Max = max(Qw); %Maximum value

%Average stagnation heat flux (non-dimensional)
Qs = HeatModel.Qs;
Qs_Max = max(Qs); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Rao (2002)
Qdot = HeatModel.Qdot;
Qdot_Max = max(Qdot); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Havey (1982)
QHavey = HeatModel.QHavey;
QHavey_Max = max(QHavey);

%Stagnation heat flux (kW/m^2); Source: Galman (1961)
QGalman = HeatModel.QGalman;
QGalman_Max = max(QGalman);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Descent-Boost Delta-V
%Descent delta-V to alter flight-path angle (km/s)
[dV_Descent] = DescentDeltaV(h_init,h_atm,rad2deg(fpa_Descent));
dV_ReCirc = abs(Skip_States(end,2) - V_RelProp);

```

```

%Descent-boost delta-V (km/s)
dV_DB = dV_Descent + dV_Boost + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Workspace Variable Definition
Trajectory_Analysis = [rad2deg(fpa_Descent),dV_Descent,dV_DB];

Deceleration_Analysis = [TangDecelG_Max,TangDecelG_Min, ...
                        NormDecelG_Max,NormDecelG_Min, ...
                        MagDecelG_Max, MagDecelG_Min];

HeatFlux_Analysis = [Qw_Max,Qs_Max,Qdot_Max,QHavey_Max,QGalman_Max];

Trajectory_States = [Skip_t,Skip_States];

```

BankManeuvers_MultiAOT.m

```

function
[Skip_t,Skip_States,Trajectory_States,RefOrb_States,Trajectory_Analysis] = ...
    BankManeuvers_MultiAOT(Vehicle_Choice,Time_Max,h_init, ...
        PSI_Ref,fpa_Descent,dV_Boost,bank_Skip)

global MU RE
WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 9 %VEHICLE SELECTION OVERRIDE
    mass = 2000; %Mass (kg)
    S_m2 = 18.5; %Planform area (m^2)
    S = S_m2/(1000^2); %Planform area (km^2)
    Cd = 0.5; %Drag coefficient
    Cl = 3.0; %Lift coefficient
else
    [Vehicle] = VehicleSpecs(Vehicle_Choice);
    mass = Vehicle.mass; %Mass (kg)

    S_m2 = Vehicle.S_m2; %Planform area (m^2)
    S = S_m2/(1000^2); %Planform area (km^2)
    Cd = Vehicle.Cd; %Drag coefficient
    Cl = Vehicle.Cl; %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
h_atm = 120; %Altitude of upper limit of sensible atmosphere (km)
lon_Ref = 0; %Initial longitude (deg)
lat_Ref = 0; %Initial geodetic latitude (deg)
fpa_Ref = 0; %Flight-path angle (deg)
bank_Ref = 0; %Reference orbit bank angle (deg)

```

```

%Converts and overwrites initial angle variables
lon_Ref      = deg2rad(lon_Ref);      lat_Ref = deg2rad(lat_Ref);
fpa_Ref      = deg2rad(fpa_Ref);      PSI_Ref = deg2rad(PSI_Ref);
fpa_Descent  = deg2rad(fpa_Descent);

%Reference orbit parameters
r_init       = h_init + RE;           %Initial radial position (km)
SMA_Ref      = 0.5*(r_init + r_init); %Reference orbit semi-major axis (km)
RefPeriod    = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_init,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank_Ref);

%Conversion of time units from minutes to seconds
Time_Max = Time_Max*(60);

SMA_Target0   = SMA_Ref;      %Target semi-major axis (km)
V_Decrement   = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel;        %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref;    %Initial guess for heading angle (rad)
IterMax       = 50;          %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
        1,1,1,1,1,0.5*RefPeriod,r_init,V_Check0(1,1), ...
        lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank_Ref);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_init + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
        ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
        ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
        asin((2*pi*(r_init)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));
IterCount = 1; %Initializes iteration counter for Secant loop

```

```

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0,Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*RefPeriod,r_init,V_Check0(ii,1), ...
            lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank_Ref);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_init + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
            (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
            ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));

        %Updated heading angle (rad)
        PSI_Check0(ii+1,1) = PSI_Ref + ...
            asin((2*pi*(r_init)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
        IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
            PSI_Check0(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t,RefOrb_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_init,V_Rel0, ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank_Ref);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Skip Entry Maneuver
%Relative states descent-boost maneuver
[V_RelBoost,fpa_RelBoost,PSI_RelBoost] = ...
    RelativeStates_Entry(h_init,dV_Boost,lon_Ref, ...
        lat_Ref,fpa_Descent,PSI_Ref);

```



```

%Trajectory simulation for skip maneuvers
[Skip_t, Skip_States] = Maneuver_MainFunction(Vehicle_Choice,1,1,1,1,1, ...
      Time_Max,r_init,V_RelBoost,lon_Ref, ...
      lat_Ref,fpa_RelBoost,PSI_RelBoost,bank_Skip);

%Skip entry trajectory states
SkipTraj_h = (Skip_States(:,1)) - RE; %Altitude (km)
SkipTraj_V = Skip_States(:,2); %Velocity (km/s)

%Re-Circularized velocity relative to rotating frame (rotating planet)
[V_RelProp,PSI_RelProp] =
RelativeStates(mass,S,Cd,Cl,SkipTraj_h(end),Skip_States(end,3), ...
      Skip_States(end,5),Skip_States(end,6),bank_Skip);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
[decel] = EntryDecel(1,mass,S,Cd,Cl, ...
      Skip_States(:,1),Skip_States(:,2), ...
      Skip_States(:,4),Skip_States(:,5));

%Tangential deceleration (g's)
TangDecelG_Max = max(decel.TangG); %Maximum value
TangDecelG_Min = min(decel.TangG); %Minimum value

%Normal deceleration (g's)
NormDecelG_Max = max(decel.NormalG); %Maximum value
NormDecelG_Min = min(decel.NormalG); %Minimum value

%Deceleration magnitude (g's)
MagDecelG_Max = max(decel.Gs); %Maximum value
MagDecelG_Min = min(decel.Gs); %Minimum value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Heat Flux
%Atmospheric density (kg/km^3)
[SkipTraj_Rho] = AtmosModel_PostAnalysis(SkipTraj_h,2);

%Maximum velocity (km/s) -- Occurs at Perigee
VMax = max(SkipTraj_V);

Emissivity = 0.8; %Emissivity
Tw_F = 0; %Wall temperature (deg F)
TMaxF = 1800; %Free-stream temperature (deg F)

%Heat transfer models
[HeatModel,Eta,T_KE] = HeatFluxModel(SkipTraj_V,SkipTraj_Rho,Emissivity, ...
      Tw_F,TMaxF,mass,S,Cd,Cl);

%Average wall heat flux (non-dimensional)
Qw = HeatModel.Qw;
Qw_Max = max(Qw); %Maximum value

```

```

%Average stagnation heat flux (non-dimensional)
Qs          = HeatModel.Qs;
Qs_Max      = max(Qs);    %Maximum value

%Stagnation heat flux (kW/m^2); Source: Rao (2002)
Qdot        = HeatModel.Qdot;
Qdot_Max    = max(Qdot); %Maximum value

%Stagnation heat flux (kW/m^2); Source: Havey (1982)
QHavey      = HeatModel.QHavey;
QHavey_Max  = max(QHavey);

%Stagnation heat flux (kW/m^2); Source: Galman (1961)
QGalman     = HeatModel.QGalman;
QGalman_Max = max(QGalman);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Descent-Boost Delta-V
%Descent delta-V to alter flight-path angle (km/s)
[dV_Descent] = DescentDeltaV(h_init,h_atm,rad2deg(fpa_Descent));

dV_ReCirc = abs(Skip_States(end,2) - V_RelProp);

%Descent-boost delta-V (km/s)
dV_DB = dV_Descent + dV_Boost + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Workspace Variable Definition
Trajectory_Analysis = [rad2deg(fpa_Descent),dV_Descent,dV_DB];

Deceleration_Analysis = [TangDecelG_Max,TangDecelG_Min, ...
                        NormDecelG_Max,NormDecelG_Min, ...
                        MagDecelG_Max, MagDecelG_Min];

HeatFlux_Analysis    = [Qw_Max,Qs_Max,Qdot_Max,QHavey_Max,QGalman_Max];

Trajectory_States    = [Skip_t,Skip_States];

```

BiElliptic.m

```
function [dV_BiElliptic,TOF,TOF1] = BiElliptic(h_Init,h_b,h_Final)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r_Init = h_Init + RE; %Initial orbit radius (km)
r_b     = h_b + RE;   %Intermediate orbit radius (km)
r_Final = h_Final + RE; %Final orbit radius (km)

%Transfer orbit semi-major axes (km)
sma_trans1 = 0.5.*(r_Init + r_b);
sma_trans2 = 0.5.*(r_b + r_Final);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Initial Orbit/Transfer Orbit #1
V_Init = sqrt(MU./r_Init);
Vt1a   = sqrt(((2*MU)./r_Init)-(MU/sma_trans1));
dV_a   = Vt1a - V_Init;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Transfer Orbits #1,2 Transition
Vt1b   = sqrt(((2*MU)./r_b)-(MU/sma_trans1));
Vt2b   = sqrt(((2*MU)./r_b)-(MU/sma_trans2));
dV_b   = Vt2b - Vt1b;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Transfer Orbit #2/Final Orbit
Vt2c   = sqrt(((2*MU)./r_Final)-(MU/sma_trans2));
V_Final = sqrt(MU./r_Final);
dV_c   = V_Final - Vt2c;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time-of-Flight and Total Delta-V Required for Bi-Elliptic Transfer
TOF1 = (pi*sqrt((sma_trans1.^3)./MU));
TOF   = (pi*sqrt((sma_trans1.^3)./MU)) + (pi*sqrt((sma_trans2.^3)./MU));
dV_BiElliptic = abs(dV_a) + abs(dV_b) + abs(dV_c); %(km/s)
```

BiElliptic_VelInput.m

```
function [dV_BiElliptic,TOF,TOF1] = ...
    BiElliptic_VelInput(h_Init,h_b,h_Final,V_Final)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r_Init = h_Init + RE; %Initial orbit radius (km)
r_b = h_b + RE; %Intermediate orbit radius (km)
r_Final = h_Final + RE; %Final orbit radius (km)

%Transfer orbit semi-major axes (km)
sma_trans1 = 0.5.*(r_Init + r_b);
sma_trans2 = 0.5.*(r_b + r_Final);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Initial Orbit/Transfer Orbit #1
V_Init = sqrt(MU./r_Init);
Vt1a = sqrt(((2*MU)./r_Init)-(MU/sma_trans1));
dV_a = Vt1a - V_Init;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Transfer Orbits #1,2 Transition
Vt1b = sqrt(((2*MU)./r_b)-(MU/sma_trans1));
Vt2b = sqrt(((2*MU)./r_b)-(MU/sma_trans2));
dV_b = Vt2b - Vt1b;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Transfer Orbit #2/Final Orbit
Vt2c = sqrt(((2*MU)./r_Final)-(MU/sma_trans2));
dV_c = V_Final - Vt2c;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time-of-Flight and Total Delta-V Required for Bi-Elliptic Transfer
TOF1 = (pi*sqrt((sma_trans1.^3)./MU));
TOF = (pi*sqrt((sma_trans1.^3)./MU)) + (pi*sqrt((sma_trans2.^3)./MU));
dV_BiElliptic = abs(dV_a) + abs(dV_b) + abs(dV_c); % (km/s)
```

DescentBoost_Molniya.m

```
clear all; clc; close all;

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maneuver Simulation
Vehicle_Choice = 1;      %Vehicle selection
Map_Choice     = 1;      %Map plotting selection
h_Target       = 502;    %Target altitude (km)
h_Init         = 1000;   %Initial altitude (km)
AltThreshold   = 150;    %Altitude threshold for interpolation (km)
PSI_Init       = 70;     %Heading angle (deg)
fpa_Descent    = -12.5;  %Flight-path angle (deg)
dV_Boost       = 0.5;    %Boost delta-V (km/s)
bank_Skip      = 0;      %Bank angle (deg)
Time_Max       = 720;    %Maximum simulation time (min)

[Skip_t1, Skip_States, Traj_States, RefOrb_States, Traj_Analysis] ...
    = BankManeuvers_MultiAOT(Vehicle_Choice, Time_Max, ...
        h_Init, PSI_Init, fpa_Descent, dV_Boost, bank_Skip);

Skip_t    = Traj_States(:,1)./60; %Time (min)
Skip_h    = Traj_States(:,2) - RE; %Altitude (km)
Skip_V    = Traj_States(:,3);      %Velocity (km/s)
Skip_lon  = Traj_States(:,4);      %Longitude (rad)
Skip_lat  = Traj_States(:,5);      %Latitude (rad)
Skip_fpa  = Traj_States(:,6);      %Flight-path angle (rad)
Skip_psi  = Traj_States(:,7);      %Heading angle (rad)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
[Vehicle] = VehicleSpecs(Vehicle_Choice);
mass = Vehicle.mass; %Mass (kg)
S_m2 = Vehicle.S_m2; %Planform area (m^2)
S     = S_m2/(1000^2); %Planform area (km^2)
Cd    = Vehicle.Cd; %Drag coefficient
Cl    = Vehicle.Cl; %Lift coefficient

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Molniya Orbit Parameters
SMA_Molniya = 26562; %Molniya orbit semi-major axis (km)

%Perigee
h_perig = h_Target; %Altitude (km)
r_perig = h_perig + RE; %Radius (km)

%Apogee
r_apog = (2*SMA_Molniya) - r_perig; %Radius (km)
h_apog = r_apog - RE; %Altitude (km)
```

```

%Molniya orbit eccentricity
ecc      = Eccentricity(r_apog,r_perig);

%Orbit velocity
V_perig = OrbitVelocity(h_perig + RE,ecc,0);   %Perigee
V_apog  = OrbitVelocity(h_apog  + RE,ecc,180); %Apogee

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Trajectory Crossings of Target Altitude
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Skip_h)
    mm = mm + 1;
    if abs(Skip_h(ii) - h_Target) < AltThreshold
        AltTGT_Crossing(mm,1) = Skip_h(ii);
        AltTGT_Crossing(mm,2) = Skip_t(ii);
        AltTGT_Crossing(mm,3) = Skip_V(ii);
        AltTGT_Crossing(mm,4) = Skip_lon(ii);
        AltTGT_Crossing(mm,5) = Skip_lat(ii);
        AltTGT_Crossing(mm,6) = Skip_fpa(ii);
        AltTGT_Crossing(mm,7) = Skip_psi(ii);
    else
        AltTGT_Crossing(mm,1:7) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(AltTGT_Crossing)
    if AltTGT_Crossing(ii) ~= 0
        mm = mm + 1;
        FlagVector_Alt(mm,1) = ii;
        WithinIdent_Alt(mm,:) = AltTGT_Crossing(ii,:);
    end
end
FlagVector_Alt = [FlagVector_Alt;0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Alt)-1
    if abs((FlagVector_Alt(ii+1) - FlagVector_Alt(ii))) > 1
        mm = mm + 1;
        AltTGT_Jump(mm,1) = ii;
    end
end
AltTGT_Jump = [0;AltTGT_Jump];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Crossing Trajectories
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(AltTGT_Jump)
    mm = mm + 1;

```

```

AltTGT_Interp(mm,:) = ...
    interp1(AltTGT_Crossing(FlagVector_Alt(AltTGT_Jump(ii-1)+1): ...
        FlagVector_Alt(AltTGT_Jump(ii)),1), ...
        AltTGT_Crossing(FlagVector_Alt(AltTGT_Jump(ii-1)+1): ...
        FlagVector_Alt(AltTGT_Jump(ii)),2:7), ...
        h_Target,'spline'); %Cubic spline interpolation
end

%Removal of negative interpolated points
AltTGT_Interp(any(AltTGT_Interp(:,1)<0,2),:) = [];

AltTGT_Vector = h_Target.*ones(length(AltTGT_Interp(:,1)),1);
AltCrossings = [AltTGT_Interp(:,1),AltTGT_Vector,AltTGT_Interp(:,2:6)];

%Removal of extremely large interpolated points
AltCrossings(any(AltCrossings(:,1)>(5*Time_Max),2),:) = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Minimum Re-Circularization Delta-V
mm = 0; %Initializes loop index at zero
nn = 1; %Initializes vector concatenation counter at one

for mm = 1:length(AltCrossings(:,1))

    %Re-Circularized velocity relative to rotating frame (rotating planet)
    [V_Rel_ReCirc(nn,1),PSI_Rel_ReCirc(nn,2)] = RelativeStates(mass,S,Cd,Cl, ...
        h_Target,AltCrossings(mm,5),AltCrossings(mm,6), ...
        AltCrossings(mm,7),bank_Skip);

    mm = mm + 1; %Update to index counter
    nn = nn + 1; %Update to solution matrix concatenation counter
end

%Re-circularization delta-V (km/s)
dV_ReCirc_Vec = abs(AltCrossings(:,3) - V_perig);

%Concatenation of re-circ. delta-V vector with crossings solutions
AltCrossings_withdV = [AltCrossings,dV_ReCirc_Vec];

%Minimum re-circularization delta-V and related states
[Min_dV,Min_Flag] = min(AltCrossings_withdV(2:end,end));
Min_States = AltCrossings_withdV(Min_Flag+1,:);
%NOTE: 'Min' search starts with Row 2 so as to prevent orbit insertion
%occurring at the first crossing of the target altitude and thus ensuring
%at least one skip in atmosphere.

%Maneuver simulation constrained by elapsed time of minimum delta-V
[Skip_t_MOD,Skip_States_MOD,Traj_States_MOD,RefOrb_States_MOD, ...
    Traj_Analysis_MOD] = BankManeuvers_MultiAOT(Vehicle_Choice, ...
        Min_States(1,1),h_Init,PSI_Init, ...
        fpa_Descent,dV_Boost,bank_Skip);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Propagation of Re-Circularized Orbit
Time_Prop = (Time_Max - Min_States(1,1))*60; %Maximum simulation time (s)
ecc        = 0; %Orbit eccentricity
r_Prop     = Skip_States_MOD(end,1); %Orbit radial position (km)
h_Prop     = r_Prop - RE; %Orbit altitude (km)
lon_Prop   = Skip_States_MOD(end,3); %Initial longitude (rad)
lat_Prop   = Skip_States_MOD(end,4); %Initial latitude (rad)
fpa_Prop   = 0; %Flight-path angle (rad)
bank_Prop  = bank_Skip; %Bank angle (deg)

if bank_Skip ~= 0
    PSI_Prop = -(min(Skip_States_MOD(:,4))); %Heading angle (rad)
else
    PSI_Prop = ((Skip_States_MOD(end,6))); %Heading angle (rad)
end

%Re-circularized orbit parameters
SMA_Prop    = 0.5*(r_Prop + r_Prop); %Semi-major axis (km)
Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_RelProp, PSI_RelProp] = RelativeStates(mass, S, Cd, Cl, h_Prop, lat_Prop, ...
                                          fpa_Prop, PSI_Prop, bank_Prop);

SMA_TargetProp = SMA_Prop;
V_Decrement    = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckP(1,1)  = V_RelProp; %Initial guess for velocity (km/s)
PSI_CheckP(1,1) = PSI_RelProp; %Initial guess for heading angle (rad)
IterMax        = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
                                                1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
                                                V_CheckP(1,1),lon_Prop,lat_Prop, ...
                                                fpa_Prop,PSI_CheckP(1,1),bank_Prop);
[r_CheckP(1,1),ApogFlag] = min(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckP(1,1) = 0.5*(r_Prop + r_CheckP(1,1));

%Iteration error (s)
GuessErrorP(1,1) = -((SMA_CheckP(1,1) - SMA_Prop)/ ...
                    ((V_CheckP(1,1) - V_Decrement) - V_CheckP(1,1)));

%Updated velocity (km/s)
V_CheckP(2,1) = (V_CheckP(1,1) - V_Decrement) - ...
                ((SMA_TargetProp - SMA_CheckP(1,1))/GuessErrorP(1,1));

%Updated heading angle (rad)
PSI_CheckP(2,1) = PSI_Prop + ...
                asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(2,1)));

```



```

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckP(1,1));
IterativeDiff_PSI_Prop(1,1) = abs(PSI_CheckP(2,1) - PSI_CheckP(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckP(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckP(ii,1) - PSI_CheckP(ii-1,1)) > 1E-10 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckP(ii,1), ...
            lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii,1),bank_Prop);

        [r_CheckP(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

        %Current iteration semi-major axis (km)
        SMA_CheckP(ii,1) = 0.5*(r_Prop + r_CheckP(ii,1));

        %Iteration error (sec)
        GuessErrorP(ii,1) = -((SMA_CheckP(ii,1) - SMA_CheckP(ii-1,1))/ ...
            (V_CheckP(ii,1) - V_CheckP(ii-1,1)));

        %Updated velocity (km/s)
        V_CheckP(ii+1,1) = V_CheckP(ii,1) - ...
            ((SMA_TargetProp - SMA_CheckP(ii,1))/GuessErrorP(ii,1));

        %Updated heading angle (rad)
        PSI_CheckP(ii+1,1) = PSI_Prop + ...
            asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp - ...
            SMA_CheckP(ii,1));
        IterativeDiff_PSI_Prop(ii,1) = abs(PSI_CheckP(ii,1) - ...
            PSI_CheckP(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t, Orbit_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Prop,r_Prop,V_CheckP(ii), ...
    lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii),bank_Prop);

```

```

%Re-defined propagated orbit states
PropOrb_t      = [Skip_t_MOD ; Skip_t_MOD(end) + Orbit_t(2:end)];
PropOrb_States = [Skip_States_MOD(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h      = PropOrb_States(:,1) - RE;      %Altitude (km)
PropOrb_V      = PropOrb_States(:,2);          %Velocity (km/s)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
    + 180),360) - 180;      %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

%Maximum inclination (deg)
MaxIncl = max(PropOrb_Lat_deg);

%Inclination change (deg)
dIncl   = MaxIncl - PSI_Init;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Delta-V
dV_Descent = Traj_Analysis_MOD(2); %Descent delta-V (km/s)
dV_ReCirc  = Min_dV;               %Re-circularization delta-V (km/s)

%Total delta-V for descent-boost skip maneuver (km/s)
dV_Total   = dV_Descent + dV_Boost + dV_ReCirc;

%Hohmann transfer delta-V (km/s)
[dV_Hohmann_perig,TOF_Hohmann_perig] =
Hohmann_VelInput(h_Init,h_perig,V_perig);
[dV_Hohmann_apog,TOF_Hohmann_apog] = Hohmann_VelInput(h_Init,h_apog, V_apog);

%Combined Hohmann transfer delta-V (km/s)
[dV_Combined_perig,TOF_Combined_perig] = ...
    Hohmann_Combined_VelInput(h_Init,h_perig,dIncl,V_perig);
[dV_Combined_apog, TOF_Combined_apog] = ...
    Hohmann_Combined_VelInput(h_Init,h_apog,dIncl,V_apog);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maneuver Time-of-Flight
%Descent-boost maneuver
TOF_Skip      = (Skip_t_MOD(end))/60;

%Hohmann transfer
TOF_Hohmann_perig = TOF_Hohmann_perig/60;
TOF_Hohmann_apog  = TOF_Hohmann_apog/60;

%Combined Hohmann transfer
TOF_Combined_perig = TOF_Combined_perig/60;
TOF_Combined_apog  = TOF_Combined_apog/60;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Re-Defines Data to Reflect Jumps in Data between 180 and -180 deg
%Reference orbit
[Lon_RefOrb, Lat_RefOrb, LonSplit_RefOrb, LatSplit_RefOrb] = ...
    CoordinateJump(RefOrb_States);

```

```

%Propagated re-circularized orbit
[Lon_PropOrb, Lat_PropOrb, LonSplit_PropOrb, LatSplit_PropOrb] = ...
    CoordinateJump(PropOrb_States);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Commands
%Conversion of time units for plotting
Skip_Time = Skip_t;
[PropOrb_Time, time_string] = TimeUtility(PropOrb_t,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) v. Time
subplot(2,2,1); box on; grid off;
hold on; plot(Skip_Time,Skip_States(:,1)-RE,'b');
hold on; plot([Skip_Time(1),Skip_Time(end)],[h_Target,h_Target],'k-.');
hold on; plot(AltCrossings(:,1),AltCrossings(:,2),'go','LineWidth',2);
xlabel('Time, min');
ylabel('Geodetic Altitude, km');

h_atm = 120; %Altitude of upper limit of sensible atmosphere (km)
hold on; plot([Skip_Time(1),Skip_Time(end)],[h_atm,h_atm],'r--');

legend('Descent-Boost Trajectory','Target Altitude','Altitude Crossings', ...
    'Upper Limit of Sensible Atmosphere','Location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) v. Time
subplot(2,2,2); box on; grid on;
hold on; plot(PropOrb_Time,PropOrb_h,'b');
xlabel('Time, min');
ylabel('Geodetic Altitude, km');

h_atm = 120; %Altitude of upper limit of sensible atmosphere (km)
hold on; plot([Skip_Time(1),Skip_Time(end)],[h_atm,h_atm],'r--');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V (km/s) v. Maneuver Type
subplot(2,2,3); box on; grid off;
dV_Bar = [dV_Descent,dV_Boost,dV_ReCirc,dV_Total, ...
    dV_Hohmann_perig,dV_Hohmann_apog];
bar(dV_Bar);
set(gca,'XTickLabel',{'Descent +','Boost +','Re-Circ.','Total Skip', ...
    'Hohmann (Perig.)','Hohmann (Apog.)'},'FontSize',8);
hold on; bar(5,dV_Hohmann_perig,'r');
hold on; bar(6,dV_Hohmann_apog,'g');
set(gca,'YTick',0:0.25:3.5);
n = get(gca,'Ytick'); set(gca,'Yticklabel',sprintf('%.2f |',n));
xlabel('Maneuver and/or Maneuver Segment','FontSize',10);
ylabel('\it\Delta V\rm, km/s','FontSize',10);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time-of-Flight v. Maneuver Type
subplot(2,2,4); box on; grid off;
TOF_Bar = [TOF_Skip,TOF_Hohmann_perig,TOF_Hohmann_apog];
bar(TOF_Bar);
set(gca,'XTickLabel',{'Descent-Boost','Hohmann (Perig.)',...
                    'Hohmann (Apog.)'},'FontSize',8);
hold on; bar(2,TOF_Hohmann_perig,'r');
hold on; bar(3,TOF_Hohmann_apog,'g');
xlabel('Maneuver','FontSize',10);
ylabel('Time-of-Flight, min','FontSize',10);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

DescentBoost_ReCirc.m

```

clear all; clc; close all;

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maneuver Simulation
Vehicle_Choice = 1; %Vehicle selection
Map_Choice     = 1; %Map plotting selection
h_Target       = 500; %Target altitude (km)
h_Init         = 500; %Initial altitude (km)
AltThreshold   = 100; %Altitude threshold for interpolation (km)
PSI_Init       = 70; %Heading angle (deg)
fpa_Descent    = -7.9; %Flight-path angle (deg)
dV_Boost       = 0.5; %Boost delta-V (km/s)
bank_Skip      = 0; %Bank angle (deg)
Time_Max       = 720; %Maximum simulation time (min)

[Skip_t1,Skip_States,Traj_States,RefOrb_States,Traj_Analysis] ...
    = BankManeuvers_MultiAOT(Vehicle_Choice,Time_Max, ...
    h_Init,PSI_Init,fpa_Descent,dV_Boost,bank_Skip);

Skip_t = Traj_States(:,1)./60; %Time (min)
Skip_h = Traj_States(:,2) - RE; %Altitude (km)
Skip_V = Traj_States(:,3); %Velocity (km/s)
Skip_lon = Traj_States(:,4); %Longitude (rad)
Skip_lat = Traj_States(:,5); %Latitude (rad)
Skip_fpa = Traj_States(:,6); %Flight-path angle (rad)
Skip_psi = Traj_States(:,7); %Heading angle (rad)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
[Vehicle] = VehicleSpecs(Vehicle_Choice);
mass = Vehicle.mass; %Mass (kg)
S_m2 = Vehicle.S_m2; %Planform area (m^2)
S = S_m2/(1000^2); %Planform area (km^2)
Cd = Vehicle.Cd; %Drag coefficient
Cl = Vehicle.Cl; %Lift coefficient

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Trajectory Crossings of Target Altitude
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Skip_h)
    mm = mm + 1;
    if abs(Skip_h(ii) - h_Target) < AltThreshold
        AltTGT_Crossing(mm,1) = Skip_h(ii);
        AltTGT_Crossing(mm,2) = Skip_t(ii);
        AltTGT_Crossing(mm,3) = Skip_V(ii);
        AltTGT_Crossing(mm,4) = Skip_lon(ii);
        AltTGT_Crossing(mm,5) = Skip_lat(ii);
        AltTGT_Crossing(mm,6) = Skip_fpa(ii);
        AltTGT_Crossing(mm,7) = Skip_psi(ii);
    else
        AltTGT_Crossing(mm,1:7) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(AltTGT_Crossing)
    if AltTGT_Crossing(ii) ~= 0
        mm = mm + 1;
        FlagVector_Alt(mm,1) = ii;
        WithinIdent_Alt(mm,:) = AltTGT_Crossing(ii,:);
    end
end
FlagVector_Alt = [FlagVector_Alt;0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Alt)-1
    if abs((FlagVector_Alt(ii+1) - FlagVector_Alt(ii))) > 1
        mm = mm + 1;
        AltTGT_Jump(mm,1) = ii;
    end
end
AltTGT_Jump = [0;AltTGT_Jump];

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Crossing Trajectories
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(AltTGT_Jump)
    mm = mm + 1;
    AltTGT_Interp(mm,:) = ...
        interp1(AltTGT_Crossing(FlagVector_Alt(AltTGT_Jump(ii-1)+1): ...
            FlagVector_Alt(AltTGT_Jump(ii)),1), ...
            AltTGT_Crossing(FlagVector_Alt(AltTGT_Jump(ii-1)+1): ...
            FlagVector_Alt(AltTGT_Jump(ii)),2:7), ...
            h_Target,'spline'); %Cubic spline interpolation
end

%Removal of negative interpolated points
AltTGT_Interp(any(AltTGT_Interp(:,1)<0,2),:) = [];

AltTGT_Vector = h_Target.*ones(length(AltTGT_Interp(:,1)),1);
AltCrossings = [AltTGT_Interp(:,1),AltTGT_Vector,AltTGT_Interp(:,2:6)];

%Removal of extremely large interpolated points
AltCrossings(any(AltCrossings(:,1)>(5*Time_Max),2),:) = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Minimum Re-Circularization Delta-V
mm = 0; %Initializes loop index at zero
nn = 1; %Initializes vector concatenation counter at one

for mm = 1:length(AltCrossings(:,1))

    %Re-Circularized velocity relative to rotating frame (rotating planet)
    [V_Rel_ReCirc(nn,1),PSI_Rel_ReCirc(nn,2)] = RelativeStates(mass,S,Cd,Cl, ...
        h_Target,AltCrossings(mm,5),AltCrossings(mm,6), ...
        AltCrossings(mm,7),bank_Skip);

    mm = mm + 1; %Update to index counter
    nn = nn + 1; %Update to solution matrix concatenation counter
end

%Re-circularization delta-V (km/s)
dV_ReCirc_Vec = abs(AltCrossings(:,3) - V_Rel_ReCirc);

%Concatenation of re-circ. delta-V vector with crossings solutions
AltCrossings_withdV = [AltCrossings,dV_ReCirc_Vec];

%Minimum re-circularization delta-V and related states
[Min_dV,Min_Flag] = min(AltCrossings_withdV(:,end));
Min_States = AltCrossings_withdV(Min_Flag,:);

%Maneuver simulation constrained by elapsed time of minimum delta-V
[Skip_t_MOD,Skip_States_MOD,Traj_States_MOD,RefOrb_States_MOD, ...
    Traj_Analysis_MOD] = BankManeuvers_MultiAOT(Vehicle_Choice, ...
        Min_States(1,1),h_Init,PSI_Init, ...
        fpa_Descent,dV_Boost,bank_Skip);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Propagation of Re-Circularized Orbit
Time_Prop = (Time_Max - Min_States(1,1))*60; %Maximum simulation time (s)
ecc        = 0;                               %Orbit eccentricity
r_Prop     = Skip_States_MOD(end,1);           %Orbit radial position (km)
h_Prop     = r_Prop - RE;                     %Orbit altitude (km)
lon_Prop   = Skip_States_MOD(end,3);           %Initial longitude (rad)
lat_Prop   = Skip_States_MOD(end,4);           %Initial latitude (rad)
fpa_Prop   = 0;                               %Flight-path angle (rad)
bank_Prop  = bank_Skip;                       %Bank angle (deg)

if bank_Skip ~= 0
    PSI_Prop = -(min(Skip_States_MOD(:,4))); %Heading angle (rad)
else
    PSI_Prop = ((Skip_States_MOD(end,6)));    %Heading angle (rad)
end

%Re-circularized orbit parameters
SMA_Prop    = 0.5*(r_Prop + r_Prop);           %Semi-major axis (km)
Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU);    %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_RelProp, PSI_RelProp] = RelativeStates(mass, S, Cd, Cl, h_Prop, lat_Prop, ...
                                           fpa_Prop, PSI_Prop, bank_Prop);

SMA_TargetProp = SMA_Prop;
V_Decrement    = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckP(1,1)  = V_RelProp;   %Initial guess for velocity (km/s)
PSI_CheckP(1,1) = PSI_RelProp; %Initial guess for heading angle (rad)
IterMax        = 50;          %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
                                                1,1,1,1,1,0.5*Period_Prop,r_Prop, ...
                                                V_CheckP(1,1),lon_Prop,lat_Prop, ...
                                                fpa_Prop,PSI_CheckP(1,1),bank_Prop);
[r_CheckP(1,1),ApogFlag] = min(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckP(1,1) = 0.5*(r_Prop + r_CheckP(1,1));

%Iteration error (s)
GuessErrorP(1,1) = -((SMA_CheckP(1,1) - SMA_Prop)/ ...
                    ((V_CheckP(1,1) - V_Decrement) - V_CheckP(1,1)));

%Updated velocity (km/s)
V_CheckP(2,1) = (V_CheckP(1,1) - V_Decrement) - ...
                ((SMA_TargetProp - SMA_CheckP(1,1))/GuessErrorP(1,1));

%Updated heading angle (rad)
PSI_CheckP(2,1) = PSI_Prop + ...
                asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(2,1)));

```

```

%Difference between calculated and target trajectory states
IterativeDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckP(1,1));
IterativeDiff_PSI_Prop(1,1) = abs(PSI_CheckP(2,1) - PSI_CheckP(1,1));

IterCount = 1; %Initializes iteration counter for Secant loop

%% Secant Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckP(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckP(ii,1) - PSI_CheckP(ii-1,1)) > 1E-10 && ...
        IterCount < IterMax

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckP(ii,1), ...

lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii,1),bank_Prop);

        [r_CheckP(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

        %Current iteration semi-major axis (km)
        SMA_CheckP(ii,1) = 0.5*(r_Prop + r_CheckP(ii,1));

        %Iteration error (sec)
        GuessErrorP(ii,1) = -((SMA_CheckP(ii,1) - SMA_CheckP(ii-1,1))/ ...
            (V_CheckP(ii,1) - V_CheckP(ii-1,1)));

        %Updated velocity (km/s)
        V_CheckP(ii+1,1) = V_CheckP(ii,1) - ...
            ((SMA_TargetProp - SMA_CheckP(ii,1))/GuessErrorP(ii,1));

        %Updated heading angle (rad)
        PSI_CheckP(ii+1,1) = PSI_Prop + ...
            asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckP(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterativeDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp - ...
            SMA_CheckP(ii,1));
        IterativeDiff_PSI_Prop(ii,1) = abs(PSI_CheckP(ii,1) - ...
            PSI_CheckP(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t, Orbit_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Prop,r_Prop,V_CheckP(ii), ...

lon_Prop,lat_Prop,fpa_Prop,PSI_CheckP(ii),bank_Prop);

```



```

%Re-defined propagated orbit states
PropOrb_t      = [Skip_t_MOD ; Skip_t_MOD(end) + Orbit_t(2:end)];
PropOrb_States = [Skip_States_MOD(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h      = PropOrb_States(:,1) - RE;      %Altitude (km)
PropOrb_V      = PropOrb_States(:,2);          %Velocity (km/s)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
                      + 180),360) - 180;      %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

%Maximum inclination (deg)
MaxIncl = max(PropOrb_Lat_deg);

%Inclination change (deg)
dIncl   = MaxIncl - PSI_Init;

%Maximum apogee (km)
[ApogMax,ApogFlag] = max(Skip_h);

%Elapsed time corresponding to maximum apogee (min)
ApogTime = (Skip_t(ApogFlag));
Apogee_Output = [ApogTime,ApogMax];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Delta-V
dV_Descent = Traj_Analysis_MOD(2); %Descent delta-V (km/s)
dV_ReCirc  = Min_dV;               %Re-circularization delta-V (km/s)

%Total delta-V for descent-boost skip maneuver (km/s)
dV_Total   = dV_Descent + dV_Boost + dV_ReCirc;

%Hohmann transfer delta-V (km/s)
[dV_Hohmann,TOF_Hohmann] = Hohmann_Geocentric(h_Init,max(Skip_h));

%Combined Hohmann transfer delta-V (km/s)
[dV_Combined,TOF_Combined] = Hohmann_Combined_dI(h_Init,max(Skip_h),dIncl);

%Bi-elliptic transfer delta-V (km/s)
[dV_BiElliptic,TOF_BiElliptic] = BiElliptic(h_Init,max(Skip_h),h_Target);

%Two-perigee Hohmann transfer delta-V (km/s)
[dV_2Perig,TOF_2Perig] = Hohmann_2Perig(h_Init,max(Skip_h));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maneuver Time-of-Flight
%Descent-boost maneuver
TOF_Skip      = (Skip_t_MOD(end))/60;

%Hohmann transfer
TOF_Hohmann   = TOF_Hohmann/60;

%Combined Hohmann transfer
TOF_Combined  = TOF_Combined/60;

```

```

%Bi-elliptic transfer
TOF_BiElliptic = TOF_BiElliptic/60;

%Two-Perigee Hohmann transfer
TOF_2Perig      = TOF_2Perig/60;

%Special Output
Output = ...
    [h_Target,    dV_Descent,    dV_Boost,dV_ReCirc,    dV_Total, ...
     dV_Combined,dV_BiElliptic,TOF_Skip,TOF_Combined,TOF_BiElliptic]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Re-Defines Data to Reflect Jumps in Data between 180 and -180 deg
%Reference orbit
[Lon_RefOrb,  Lat_RefOrb,  LonSplit_RefOrb,  LatSplit_RefOrb] = ...
    CoordinateJump(RefOrb_States);

%Propagated re-circularized orbit
[Lon_PropOrb, Lat_PropOrb, LonSplit_PropOrb, LatSplit_PropOrb] = ...
    CoordinateJump(PropOrb_States);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Commands
%Conversion of time units for plotting
Skip_Time = Skip_t;
[PropOrb_Time, time_string] = TimeUtility(PropOrb_t,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) v. Time
subplot(2,2,1); box on; grid off;
hold on; plot(Skip_Time,Skip_States(:,1)-RE,'b');
hold on; plot([Skip_Time(1),Skip_Time(end)], [h_Target,h_Target], 'k-.');
hold on; plot(AltCrossings(:,1),AltCrossings(:,2), 'go', 'LineWidth',2);
xlabel('Time, min');
ylabel('Altitude, km');

h_atm = 120; %Altitude of upper limit of sensible atmosphere (km)
hold on; plot([Skip_Time(1),Skip_Time(end)], [h_atm,h_atm], 'r--');

legend('Descent-Boost Trajectory','Target Altitude','Altitude Crossings', ...
    'Upper Atmosphere Limit','Location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) v. Time
subplot(2,2,2); box on; grid off;
hold on; plot(PropOrb_Time,PropOrb_h,'b');
xlabel('Time, min');
ylabel('Altitude, km');

h_atm = 120; %Altitude of upper limit of sensible atmosphere (km)
hold on; plot([Skip_Time(1),Skip_Time(end)], [h_atm,h_atm], 'r--');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V (km/s) v. Maneuver Type
subplot(2,2,3); box on; grid off;
dV_Bar = [dV_Descent, dV_Boost, dV_ReCirc, dV_Total, ...
          dV_Combined,dV_BiElliptic];
bar(dV_Bar);
set(gca,'XTickLabel',{'Descent +','Boost +','Inject =','Total Skip', ...
                    'Combined','Bi-Elliptic'},'FontSize',8);
hold on; bar(5,dV_Combined,'r');
hold on; bar(6,dV_BiElliptic,'g');
% hold on; bar(7,dV_Hohmann,'m');
% hold on; bar(8,dV_2Perig,'c');
set(gca,'YTick',0:0.25:2.0);
n = get(gca,'Ytick'); set(gca,'Yticklabel',sprintf('%.2f |',n));
xlabel('Maneuver and/or Maneuver Segment','FontSize',10);
ylabel('\it\DeltaV\rm, km/s','FontSize',10);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,4); box on; grid off;
hold on;
h_Skip = cellfun(@plot,LonSplit_PropOrb, LatSplit_PropOrb);
hold on;
h_Ref = cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);

set(h_Skip,'LineStyle','--','Color','r');
set(h_Ref, 'LineStyle','-','Color','b');

xlim([-180 180]); ylim([-90 90]);
% xlim([0 90]); ylim([30 70]);
% xlim([floor(Lon_Target)-30, ceil(Lon_Target)+30]);
% ylim([floor(Lat_Target)-20, ceil(Lat_Target)+20]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

if Map_Choice == 1
    % hold on; %Plate Carree world map projection
    % landareas = shaperead('landareas.shp','UseGeoCoords',true);
    % geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

elseif Map_Choice == 2
    hold on; %Plate Carree world map projection
    landareas = shaperead('landareas.shp','UseGeoCoords',true);
    geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time-of-Flight v. Maneuver Type
figure; subplot(2,2,1); box on; grid off;
TOF_Bar = [TOF_Skip,TOF_Combined,TOF_BiElliptic];
bar(TOF_Bar);
set(gca,'XTickLabel',{'Descent-Boost','Combined',...
                    'Bi-Elliptic'},'FontSize',8);
hold on; bar(2,TOF_Combined,'r');
hold on; bar(3,TOF_BiElliptic,'g');
% hold on; bar(4,TOF_Hohmann,'m');
% hold on; bar(5,TOF_2Perig,'c');
set(gca,'YTick',0:50:600);
% n = get(gca,'Ytick'); set(gca,'Yticklabel',sprintf('%.2f | ',n));
xlabel('Maneuver','FontSize',10);
ylabel('Time-of-Flight, min','FontSize',10);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

Hohmann_Analysis_Molniya.m

```

clear all; clc; close all;

global RE

WGS84Constants; %Loads global constants from external m-file

Simulation_Choice = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Simulation_Choice == 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Skip Maneuver Analysis
perig_Skip = 84.7675; %Perigee altitude (km)
PSI_Skip   = 58.8;   %Initial heading angle (deg)
h_Skip     = 1000;   %Initial altitude (km)
Bank_Skip  = -90;    %Bank angle (deg)
[Traj_Analysis,Incl_Analysis,Combined_Analysis] = ...
    BankManeuvers_fxnAltTGT(1,h_Skip,perig_Skip,PSI_Skip,Bank_Skip);

Skip_Time = Traj_Analysis(1,1); %Elapsed time for single skip maneuver (sec)
dI_Skip   = Incl_Analysis(1,3); %Inclination change for skip maneuver (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Molniya Orbit Parameters
SMA_Molniya = 26562; %Molniya orbit semi-major axis (km)

```

```

%Perigee
h_perig = Traj_Analysis(1,3);           %Altitude (km)
r_perig = h_perig + RE;                 %Radius (km)

%Apogee
r_apog = (2*SMA_Molniya) - r_perig; %Radius (km)
h_apog = r_apog - RE;                 %Altitude (km)

%Molniya orbit eccentricity
ecc = Eccentricity(r_apog,r_perig);

%Orbit velocity
V_perig = OrbitVelocity(h_perig + RE,ecc,0); %Perigee
V_apog = OrbitVelocity(h_apog + RE,ecc,180); %Apogee

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Skip Maneuver Delta-V
dV_Maneuver = Traj_Analysis(1,5); %Delta-V for skip w/o re-circ (km/s)
V_EndSkip = Traj_Analysis(1,4); %Velocity at skip apogee (km/s)
dV_Insert = abs(V_perig - V_EndSkip); %Molniya insertion delta-V (km/s)

%Total delta-V for skip maneuver (km/s)
dV_Skip = dV_Maneuver + dV_Insert;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Hohmann Transfer Analysis
h_init = [300:1:5000]'; %Initial altitude (km)
dI = (linspace(0.5,30,length(h_init)))'; %Inclination change (deg)

[dV_perig,TOF_perig] = Hohmann_VelInput(h_init,h_perig,V_perig);
[dV_apog, TOF_apog] = Hohmann_VelInput(h_init,h_apog, V_apog);

%Combined Hohmann transfer simulations
[dV_Combined_0300,TOF_Combined_0300] = ...
    Hohmann_Combined_VelInput(300,h_perig,dI,V_perig);
[dV_Combined_0504,TOF_Combined_0504] = ...
    Hohmann_Combined_VelInput(h_perig,h_perig,dI,V_perig);
[dV_Combined_1000,TOF_Combined_1000] = ...
    Hohmann_Combined_VelInput(1000,h_perig,dI,V_perig);
[dV_Combined_5000,TOF_Combined_5000] = ...
    Hohmann_Combined_VelInput(5000,h_perig,dI,V_perig);

TOF_Combined_0300 = TOF_Combined_0300.*ones(length(dV_Combined_0300),1);
TOF_Combined_0504 = TOF_Combined_0504.*ones(length(dV_Combined_0504),1);
TOF_Combined_1000 = TOF_Combined_1000.*ones(length(dV_Combined_1000),1);
TOF_Combined_5000 = TOF_Combined_5000.*ones(length(dV_Combined_5000),1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Initial Altitude (km)
subplot(2,2,1); box on; grid off;
hold on; plot(h_init,dV_perig,'b-');

```

```

hold on; plot(h_init,dV_apog,'r-');
hold on; plot(h_Skip,dV_Skip,'kd','LineWidth',2);
xlabel('Initial Altitude, km');
ylabel('\it\Delta V\rm, km/s');
legend('Perigee Transfer','Apogee Transfer', ...
       'Skip Entry, \it\sigma\rm = -90^o','Location','SouthWest');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Inclination Change (deg)
subplot(2,2,2); box on; grid off;
hold on; plot(dI,dV_Combined_0300,'k-');
hold on; plot(dI,dV_Combined_0504,'b-');
hold on; plot(dI,dV_Combined_1000,'r-');
hold on; plot(dI,dV_Combined_5000,'g-');
hold on; plot(dI_Skip,dV_Skip,'kd','LineWidth',2);
xlabel('Inclination Change, deg');
ylabel('\it\Delta V\rm, km/s');
legend('\ith_i\rm = 300 km', ...
       ['\ith_i\rm = ',num2str(floor(h_perig)),' km'], ...
       '\ith_i\rm = 1000 km','\ith_i\rm = 5000 km', ...
       'Skip Entry, \it\sigma\rm = -90^o','Location','NorthWest');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Time-of-Flight
subplot(2,2,3); box on; grid off;
hold on; plot(TOF_perig./60,dV_perig,'b-');
hold on; plot(TOF_apog./60,dV_apog,'r-');
hold on; plot(Skip_Time./60,dV_Skip,'kd','LineWidth',2);
xlabel('Time-of-Flight to Orbit Injection, min');
ylabel('\it\Delta V\rm, km/s');
legend('Perigee Transfer','Apogee Transfer', ...
       'Skip Entry, \it\sigma\rm = -90^o','Location','SouthWest');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Time-of-Flight
subplot(2,2,4); box on; grid off;
hold on; plot3(TOF_Combined_0300./60,dI,dV_Combined_0300,'k-');
hold on; plot3(TOF_Combined_0504./60,dI,dV_Combined_0504,'b-');
hold on; plot3(TOF_Combined_1000./60,dI,dV_Combined_1000,'r-');
hold on; plot3(TOF_Combined_5000./60,dI,dV_Combined_5000,'g-');
hold on; plot3(Skip_Time./60,dI_Skip,dV_Skip,'kd','LineWidth',2);
xlabel('Time-of-Flight to Orbit Injection, min');
ylabel('Inclination Change, deg');
zlabel('\it\Delta V\rm, km/s');
legend('\ith_i\rm = 300 km', ...
       ['\ith_i\rm = ',num2str(floor(h_perig)),' km'], ...
       '\ith_i\rm = 1000 km','\ith_i\rm = 5000 km', ...
       'Skip Entry, \it\sigma\rm = -90^o','Location','NorthEastOutSide');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif Simulation_Choice == 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Hohmann Transfer Analysis
h_init   = [300:1:5000]'; %Initial altitude (km)
h_perig  = 504;           %Molniya perigee altitude (km)
h_apog   = 39834;         %Molniya apogee altitude (km)
dI       = (linspace(1,30,length(h_init)))'; %Inclination change (deg)

%Molniya orbit eccentricity
ecc      = Eccentricity(h_apog + RE,h_perig + RE);

%Orbit velocity
V_perig  = OrbitVelocity(h_perig + RE,ecc,0); %Perigee
V_apog   = OrbitVelocity(h_apog + RE,ecc,180); %Apogee

% Hohmann transfer simulation
[dV_perig,TOF_perig] = Hohmann_VelInput(h_init,h_perig,V_perig);
[dV_apog, TOF_apog]  = Hohmann_VelInput(h_init,h_apog, V_apog);

%Combined Hohmann transfer simulations
[dV_Combined_0300,TOF_Combined_0300] = ...
    Hohmann_Combined_VelInput(300,h_perig,dI,V_perig);
[dV_Combined_0504,TOF_Combined_0504] = ...
    Hohmann_Combined_VelInput(504,h_perig,dI,V_perig);
[dV_Combined_1000,TOF_Combined_1000] = ...
    Hohmann_Combined_VelInput(1000,h_perig,dI,V_perig);
[dV_Combined_5000,TOF_Combined_5000] = ...
    Hohmann_Combined_VelInput(5000,h_perig,dI,V_perig);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Initial Altitude (km)
subplot(1,2,1); box on; grid off;
hold on; plot(h_init,dV_perig,'b-');
hold on; plot(h_init,dV_apog,'r-');
xlabel('Initial Altitude, km');
ylabel('\it\DeltaV\rm, km/s');
legend('Perigee Transfer','Apogee Transfer','Location','SouthWest');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Delta-V (km/s) v. Inclination Change (deg)
subplot(1,2,2); box on; grid off;
hold on; plot(dI,dV_Combined_0300,'k-');
hold on; plot(dI,dV_Combined_0504,'b-');
hold on; plot(dI,dV_Combined_1000,'r-');
hold on; plot(dI,dV_Combined_5000,'g-');
xlabel('Inclination Change, deg');
ylabel('\it\DeltaV\rm, km/s');
legend('\ith_i\rm = 300 km', '\ith_i\rm = 504 km', ...
    '\ith_i\rm = 1000 km', '\ith_i\rm = 5000 km','Location','NorthWest');

end

```

Hohmann_Combined.m

```
function [dV_Combined,TOF] = Hohmann_Combined(h1,h2,i1_deg,i2_deg)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r1 = h1 + RE; r2 = h2 + RE;

TOF    = pi*sqrt(((r1 + r2).^3)./(8*MU)); %Time-of-flight (sec)
sma_t  = (r1 + r2)./2;                  %Semi-major axis (km)
e_t    = -MU./(2.*sma_t);               %Specific mech. energy (km^2/s^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Inclination-Change Parameters
i1 = deg2rad(i1_deg); i2 = deg2rad(i2_deg);
dI = i2 - i1;

%Estimation method
R  = r2./r1;
s  = (1./dI).*atan(sin(dI)./((R.^(3/2)) + cos(dI)));
dI_init = s.*dI;
dI_final = (1 - s).*dI;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
Vc1 = sqrt(MU./r1);
Vp  = sqrt(2.*((MU./r1) + e_t));

dV_1 = sqrt((Vc1.^2) + (Vp.^2) - (2.*Vc1.*Vp.*cos(dI_init)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
Vc2 = sqrt(MU./r2);
Va  = sqrt(2.*((MU./r2) + e_t));

dV_2 = sqrt((Vc2.^2) + (Va.^2) - (2.*Vc2.*Va.*cos(dI_final)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Combined = dV_1 + dV_2;
```


Hohmann_Combined_dI.m

```
function [dV_Combined,TOF] = Hohmann_Combined_dI(h1,h2,dI_deg)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r1 = h1 + RE; r2 = h2 + RE;

TOF    = pi*sqrt(((r1 + r2).^3)./(8*MU)); %Time-of-flight (sec)
sma_t  = (r1 + r2)./2;                  %Semi-major axis (km)
e_t    = -MU./(2.*sma_t);               %Specific mech. energy (km^2/s^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Inclination Change Parameters
dI = deg2rad(dI_deg);

%Estimation method
R = r2./r1;
s = (1./dI).*atan(sin(dI)./((R.^(3/2)) + cos(dI)));
dI_init = s.*dI;
dI_final = (1 - s).*dI;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
Vc1 = sqrt(MU./r1);
Vp = sqrt(2.*((MU./r1) + e_t));
dV1 = sqrt((Vc1.^2) + (Vp.^2) - (2.*Vc1.*Vp.*cos(dI_init)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
Vc2 = sqrt(MU./r2);
Va = sqrt(2.*((MU./r2) + e_t));
dV2 = sqrt((Vc2.^2) + (Va.^2) - (2.*Vc2.*Va.*cos(dI_final)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Combined = dV1 + dV2;
```

Hohmann_Combined_VelInput.m

```
function [dV_Combined,TOF] = Hohmann_Combined_VelInput(h1,h2,dI_deg,V2)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r1 = h1 + RE; r2 = h2 + RE;

TOF    = pi*sqrt(((r1 + r2).^3)./(8*MU)); %Time-of-flight (sec)
sma_t  = (r1 + r2)./2;                  %Semi-major axis (km)
e_t    = -MU./(2.*sma_t);                %Specific mech. energy (km^2/s^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Inclination Change Parameters
dI = deg2rad(dI_deg);

%Estimation method
R   = r2./r1;
s   = (1./dI).*atan(sin(dI)./((R.^(3/2)) + cos(dI)));
dI_init = s.*dI;
dI_final = (1 - s).*dI;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
V1 = sqrt(MU./r1);
Vt1 = sqrt(2.*((MU./r1) + e_t));
dV1 = sqrt((V1.^2) + (Vt1.^2) - (2.*V1.*Vt1.*cos(dI_init)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
%Note: The velocity 'V2' is a function input representing either:
% (a) Circular orbit velocity
% (b) Apogee orbit velocity
% (c) Perigee orbit velocity
Vt2 = sqrt(2.*((MU./r2) + e_t));
dV2 = sqrt((V2.^2) + (Vt2.^2) - (2.*V2.*Vt2.*cos(dI_final)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Combined = dV1 + dV2;
```

Hohmann_Geocentric.m

```
function [dV_Total,TOF,ecc_t,sma_t] = Hohmann_Geocentric(h1,h2)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Conversion from Altitude to Geocentric Radius
r1 = h1 + RE; r2 = h2 + RE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Transfer Orbit
ecc_t = abs(((r2 - r1)./(r2 + r1))); %Eccentricity
sma_t = (r1 + r2)./2; %Semi-major axis (km)
e_t = -MU./(2.*sma_t); %Specific mech. energy (km^2/s^2)
TOF = pi.*sqrt(((r1 + r2).^3)./(8.*MU)); %Time-of-flight (s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
V1 = sqrt(MU./r1); %Circular orbit velocity (km/s)
Vt1 = sqrt(2.*((MU./r1) + e_t)); %Transfer orbit velocity at r1 (km/s)
dV1 = abs(V1 - Vt1); %Delta-V to enter transfer orbit (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
V2 = sqrt(MU./r2); %Circular orbit velocity (km/s)
Vt2 = sqrt(2.*((MU./r2) + e_t)); %Transfer orbit velocity at r2 (km/s)
dV2 = abs(V2 - Vt2); %Delta-V to re-circularize at r2 (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Total = dV1 + dV2;
```

Hohmann_Geodetic.m

```
function [dV_Total,TOF,ecc_t,sma_t] = Hohmann_Geodetic(h_gd1,h_gd2)

global MU RE FlatE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Conversion from Geodetic to Geocentric Coordinates
[r1, lat1] = Geodetic2Geocentric(h_gd1,0,RE,FlatE);
[r2, lat2] = Geodetic2Geocentric(h_gd2,0,RE,FlatE);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transfer Orbit
ecc_t = abs(((r2 - r1)./(r2 + r1))); %Eccentricity
sma_t = (r1 + r2)./2; %Semi-major axis (km)
e_t = -MU./(2.*sma_t); %Specific mech. energy (km^2/s^2)
TOF = pi.*sqrt(((r1 + r2).^3)./(8.*MU)); %Time-of-flight (s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Velocity Parameters for Orbit #1
V1 = sqrt(MU./r1); %Circular orbit velocity (km/s)
Vt1 = sqrt(2.*((MU./r1) + e_t)); %Transfer orbit velocity at r1 (km/s)
dV1 = abs(V1 - Vt1); %Delta-V to enter transfer orbit (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Velocity Parameters for Orbit #2
V2 = sqrt(MU./r2); %Circular orbit velocity (km/s)
Vt2 = sqrt(2.*((MU./r2) + e_t)); %Transfer orbit velocity at r2 (km/s)
dV2 = abs(V2 - Vt2); %Delta-V to re-circularize at r2 (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Total Delta-V Required for Combined Hohmann Transfer
dV_Total = dV1 + dV2;
```

Hohmann_SkipReCirc.m

```
function [dV_Total,TOF,ecc_t,sma_t] = Hohmann_SkipReCirc(V1,r1,r2)

global MU

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Transfer Orbit
ecc_t = abs(((r2 - r1)./(r2 + r1))); %Eccentricity
sma_t = (r1 + r2)./2; %Semi-major axis (km)
e_t = -MU./(2.*sma_t); %Specific mech. energy (km^2/s^2)
TOF = pi.*sqrt(((r1 + r2).^3)./(8.*MU)); %Time-of-flight (s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
Vt1 = sqrt(2.*((MU./r1) + e_t)); %Transfer orbit velocity at r1 (km/s)
dV1 = abs(V1 - Vt1); %Delta-V to enter transfer orbit (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
V2 = sqrt(MU./r2); %Circular orbit velocity (km/s)
Vt2 = sqrt(2.*((MU./r2) + e_t)); %Transfer orbit velocity at r2 (km/s)
dV2 = abs(V2 - Vt2); %Delta-V to recircularize at r2 (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Total = dV1 + dV2;
```

Hohmann_VelInput.m

```
function [dV_Total,TOF] = Hohmann_VelInput(h1,h2,V2)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculation of Transfer Orbit Parameters
r1 = h1 + RE; r2 = h2 + RE;

TOF    = pi*sqrt(((r1 + r2).^3)./(8*MU)); %Time-of-flight (sec)
sma_t  = (r1 + r2)./2;                  %Semi-major axis (km)
e_t    = -MU./(2.*sma_t);                %Specific mech. energy (km^2/s^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #1
V1      = sqrt(MU./r1);                  %Circular orbit velocity (km/s)
Vt1     = sqrt(2.*((MU./r1) + e_t)); %Transfer orbit velocity at r1 (km/s)
dV1     = abs(V1 - Vt1);                 %Delta-V to enter transfer orbit (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Velocity Parameters for Orbit #2
%Note: The velocity 'V2' is a function input representing either:
% (a) Circular orbit velocity
% (b) Apogee orbit velocity
% (c) Perigee orbit velocity
Vt2     = sqrt(2.*((MU./r2) + e_t)); %Transfer orbit velocity at r2 (km/s)
dV2     = abs(V2 - Vt2);                 %Delta-V to recircularize at r2 (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Total Delta-V Required for Combined Hohmann Transfer
dV_Total = dV1 + dV2;
```

PlanarManeuvers.m

```
% function [Trajectory_Analysis] = PlanarManeuvers(Target_Choice,Xing)
clear all; clc; close all;

global MU RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape

WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Selection
Vehicle_Choice = 1;
Target_Choice = 5;
Xing = 24;
VCoeff = .965; %Fraction coefficient to modify velocity guess

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver; Lon_Target = Lon_Denver; dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow; Lon_Target = Lon_Glasgow; dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny; Lon_Target = Lon_Grozny; dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow; Lon_Target = Lon_Moscow; dLat = 5;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti; Lon_Target = Lon_Ponti; dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong; Lon_Target = Lon_Pyong; dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk; Lon_Target = Lon_Reyk; dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran; Lon_Target = Lon_Tehran; dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo; Lon_Target = Lon_Tokyo; dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil; Lon_Target = Lon_Brasil; dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos; Lon_Target = Lon_Buenos; dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra; Lon_Target = Lon_Canberra; dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape; Lon_Target = Lon_Cape; dLat = 4;
end
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 9      %VEHICLE SELECTION OVERRIDE
    mass = 2000;           %Mass (kg)
    S_m2 = 18.5;           %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = 0.5;           %Drag coefficient
    Cl    = 3.0;           %Lift coefficient
else
    [Vehicle] = VehicleSpecs(Vehicle_Choice);
    mass = Vehicle.mass;   %Mass (kg)
    S_m2 = Vehicle.S_m2;   %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Vehicle.Cd;    %Drag coefficient
    Cl    = Vehicle.Cl;    %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
Time_Max = 1;      %Maximum simulation time (days)
ecc_Ref   = 0;      %Orbit eccentricity
h_Ref     = 1000;   %Orbit geodetic altitude (km)
lon_Ref   = 0;      %Initial longitude (deg)
lat_Ref   = 0;      %Initial geodetic latitude (deg)
fpa_Ref   = 0;      %Flight-path angle (deg)
PSI_Ref   = 70;     %Heading angle (deg)
bank      = 0;      %Bank angle (deg)

%Converts and overwrites initial angle variables from (deg) to (rad)
lon_Ref = deg2rad(lon_Ref); lat_Ref = deg2rad(lat_Ref);
fpa_Ref = deg2rad(fpa_Ref); PSI_Ref = deg2rad(PSI_Ref);

%Reference orbit parameters
r_Ref     = h_Ref + RE; %Radial position (km)
SMA_Ref    = 0.5*(r_Ref + r_Ref); %Semi-major axis (km)
RefPeriod  = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)
V_Ref     = sqrt(MU*((2/r_Ref) - (1/SMA_Ref))); %Orbit velocity (km/s)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_Ref,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank);

%Conversion of time units from days to seconds
Time_Max = Time_Max*(24)*(60)*(60);

SMA_Target0 = SMA_Ref; %Target semi-major axis for iteration (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel; %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

```



```

%% First Iteration
%Trajectory simulation [0:t: 0.5*RefPeriod]
[Traj_t0, Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
        1,1,1,1,1,0.5*RefPeriod,r_Ref,V_Check0(1,1), ...
        lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_Ref + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
        ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
        ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
        asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount0 = 1; %Initializes iteration counter for Newton-Raphson loop

%% Newton-Raphson Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-20 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-20

        %Trajectory simulation [0:t: 0.5*RefPeriod]
        [Traj_t0, Traj_States0] = Maneuver_MainFunction(Vehicle_Choice, ...
                1,1,1,1,1,0.5*RefPeriod,r_Ref,V_Check0(ii,1), ...
                lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_Ref + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
                (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
                ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));
    end
end

```

```

%Updated heading angle (rad)
PSI_Check0(ii+1,1) = PSI_Ref + ...
    asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

%Difference between calculated and target trajectory states
IterDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
IterDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1));

ii = ii + 1; %Update to row-index counter
IterCount0 = IterCount0 + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Orbital velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t, RefOrb_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Ref,V_Rel0, ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Manipulation of Reference Orbit Trajectory Solutions
r_Data = RefOrb_States(:,1); %Radial position (km)
h_Data = r_Data - RE; %Altitude (km)
Lon_Data = RefOrb_States(:,3); %Longitude (rad)
Lat_Data = RefOrb_States(:,4); %Geocentric latitude (rad)

%Transforms longitude from (0 <= lon < 360) to (-180 < lon <= 180)
Lon_Data = rem((rad2deg(Lon_Data) + 180),360) - 180;

%Converts geodetic latitude from radians to degrees
Lat_Data = rad2deg(Lat_Data);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Trajectory Crossings of Target Latitude
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Lon_Data)
    mm = mm + 1;
    if abs(Lat_Data(ii) - Lat_Target) < 10
        LatCrossing(mm,1) = RefOrb_t(ii); %Time (sec)
        LatCrossing(mm,2) = h_Data(ii); %Altitude (km)
        LatCrossing(mm,3) = Lon_Data(ii); %Longitude (deg)
        LatCrossing(mm,4) = Lat_Data(ii); %Geocentric latitude (deg)
    else
        LatCrossing(mm,1:4) = 0; %Arbitrary value for non-crossings
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Latitude Crossings
mm = 0; %Initializes vector concatenation counter at zero

for ii = 1:length(LatCrossing)
    if LatCrossing(ii) ~= 0
        mm = mm + 1;
        FlagVector(mm,1) = ii;
        CrossingIdent(mm,:) = LatCrossing(ii,:);
    end
end
FlagVector = [FlagVector;0]; %Indices corresponding to latitude crossings

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Latitude Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector)-1
    if abs((FlagVector(ii+1) - FlagVector(ii))) > 1
        mm = mm + 1;
        CrossingJump(mm,1) = ii;
    end
end
CrossingJump = [0;CrossingJump]; %Indices of jumps in latitude crossings

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Latitude Crossing Trajectories
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(CrossingJump)
    mm = mm + 1;
    CrossInterp(mm,:) = ...
        interp1(LatCrossing(FlagVector(CrossingJump(ii-1)+1): ...
            FlagVector(CrossingJump(ii)),4), ...
            LatCrossing(FlagVector(CrossingJump(ii-1)+1): ...
            FlagVector(CrossingJump(ii)),1:3), ...
            Lat_Target,'spline'); %Cubic spline interpolation
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Identifies whether Crossing is East or West of Target Longitude
mm = 0; %Initializes vector concatenation counter for East crossings
nn = 0; %Initializes vector concatenation counter for West crossings
for ii = 1:length(CrossInterp(:,3))
    if CrossInterp(ii,3) > Lon_Target && CrossInterp(ii,3) < 180
        mm = mm + 1;
        CrossingEast(mm,1) = CrossInterp(ii,1); %Time (sec)
        CrossingEast(mm,2) = CrossInterp(ii,3); %Longitude (deg)
    elseif CrossInterp(ii,3) < Lon_Target && CrossInterp(ii,3) > -180
        nn = nn + 1;
        CrossingWest(nn,1) = CrossInterp(ii,1); %Time (sec)
        CrossingWest(nn,2) = CrossInterp(ii,3); %Longitude (deg)
    end
end

EastFlag = 1.*ones(length(CrossingEast),1); %Flag indicating 'East' crossing
WestFlag = 2.*ones(length(CrossingWest),1); %Flag indicating 'West' crossing

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Latitude Crossing Data
%Difference between interpolated and target longitudes
dLonEast = abs(CrossingEast(:,2) - Lon_Target);
dLonWest = abs(CrossingWest(:,2) - Lon_Target);

%Number of perturbed orbits ('fix' truncation yields integer values)
OrbNumEast = fix(CrossingEast(:,1)./RefPeriod);
OrbNumWest = fix(CrossingWest(:,1)./RefPeriod);

%Array components: Time, longitude, longitude difference, number of orbits
Crossings = [CrossingEast, dLonEast, OrbNumEast, EastFlag; ...
             CrossingWest, dLonWest, OrbNumWest, WestFlag];

%Removal of rows with zero reference orbits
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Crossings)
    if Crossings(ii,4) ~= 0
        mm = mm + 1;
        Crossings_States(mm,:) = Crossings(ii,:);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Perturbed Orbit Parameters
%Delta-period between crossings and target per orbit (sec/orbit)
dPeriod = Crossings_States(:,3).*(1/15).*(3600).*(1./Crossings_States(:,4));

%Perturbed orbit periods (sec)
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Crossings_States)
    if Crossings_States(ii,end) == 1 %East Crossing (Increase SMA)
        mm = mm + 1;
        Period_Skip0(mm,:) = [RefPeriod + dPeriod(ii,1), ...
                               Crossings_States(ii,end)];

    elseif Crossings_States(ii,end) == 2 %West Crossing (Decrease SMA)
        mm = mm + 1;
        Period_Skip0(mm,:) = [RefPeriod - dPeriod(ii,1), ...
                               Crossings_States(ii,end)];
    end
end

%Perturbed orbit semi-major axes (km)
SMA_Skip0 = [(MU.*((Period_Skip0(:,1)./(2*pi)).^2)).^(1/3), ...
             Crossings_States(:,end)]];

%Array components: Time, longitude, longitude difference,
%                  number of orbits, skip period, skip SMA
Crossings_FullStates = [Crossings_States(:,1:4),Period_Skip0(:,1),SMA_Skip0];

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Transformation of 'Impacting' Perturbed Orbits
mm = 0; nn = 0; %Initializes vector concatenation counters at zero
for ii = 1:length(Crossings_FullStates(:,1))
    if Crossings_FullStates(ii,6) < 7000 %(km)
        mm = mm + 1;
        GroundImpact(mm,:) = Crossings_FullStates(ii,:);
    elseif Crossings_FullStates(ii,6) > 7000 %(km)
        nn = nn + 1;
        NoImpact(nn,:) = Crossings_FullStates(ii,:);
    end
end

%Updated longitude difference, crossing time, and number of reference orbits
dLonWest_Update0 = 360 - (GroundImpact(:,3));
dLonWest_Time0 = GroundImpact(:,1) + (dLonWest_Update0/360);
OrbNumWest_Update0 = fix(dLonWest_Time0(:,1)./RefPeriod);

%Removal of rows with zero reference orbits
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(OrbNumWest_Update0)
    if OrbNumWest_Update0(ii,1) ~= 0
        mm = mm + 1;
        dLonWest_Update(mm,:) = dLonWest_Update0(ii,:);
        OrbNumWest_Update(mm,:) = OrbNumWest_Update0(ii,:);
    end
end

%Updated delta-periods (sec)
dPeriod_Update = dLonWest_Update(:,1).*(1/15).*(60).*(60).* ...
    (1./OrbNumWest_Update(:,1));

%Updated perturbed orbit periods (sec)
Period_Skip_Update = RefPeriod + dPeriod_Update;

%Assignment of 'East' crossing flag since maneuver is now 'ascending'
EastFlag_Update = 1.*ones(length(dLonWest_Update),1);

%Updated perturbed orbit semi-major axes (km)
SMA_Skip_Update = (MU.*((Period_Skip_Update(:,1)./(2*pi)).^2)).^(1/3);

%Unsorted perturbed orbit parameters
SMA_Skip_UnSort = [NoImpact(:,5), NoImpact(:,6), NoImpact(:,4), ...
    NoImpact(:,2), NoImpact(:,end); ...
    Period_Skip_Update(:,1), SMA_Skip_Update(:,1), ...
    OrbNumWest_Update(:,1), GroundImpact(:,2), ...
    EastFlag_Update(:,1)];

%Sorting of perturbed orbit parameters according to crossing flag
[SMA_Sort,I] = sort(SMA_Skip_UnSort(:,5));
SMA_Skip = SMA_Skip_UnSort(I,:);

%Removal of negative perturbed periods
SMA_Skip(any(SMA_Skip(:,1)<0,2),:) = [];

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Maneuver (Descent or Ascent) Velocity
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(SMA_Skip)
    mm = mm + 1;
    if SMA_Skip(ii,end) == 1 %East Crossing, Reference Orbit = Perigee
        SMA_Target(mm,:) = [SMA_Skip(ii,2),SMA_Skip(ii,end)];
        r_Perig(mm,1) = r_Ref;
        r_Apog(mm,1) = (2.*SMA_Target(mm,1)) - r_Perig(mm,1);
        HalfPeriod(mm,1) = (0.5).*SMA_Skip(ii,1);
        V_Initial(mm,1) = sqrt((2.*MU.*r_Apog(mm,1))./ ...
            (r_Perig(mm,1).*(r_Apog(mm,1) + r_Perig(mm,1))));
    elseif SMA_Skip(ii,end) == 2 %West Crossing, Reference Orbit = Apogee
        SMA_Target(mm,:) = [SMA_Skip(ii,2),SMA_Skip(ii,end)];
        r_Apog(mm,1) = r_Ref;
        r_Perig(mm,1) = (2.*SMA_Target(mm,1)) - r_Apog(mm,1);
        HalfPeriod(mm,1) = (0.5).*SMA_Skip(ii,1);
        V_Initial(mm,1) = sqrt((2.*MU.*r_Perig(mm,1))./ ...
            (r_Apog(mm,1).*(r_Apog(mm,1) + r_Perig(mm,1))));
    end
end

V_Decrement = 1 - 0.9999; %Decrement value for velocity
V_Check(1,1) = VCoeff.*V_Initial(Xing,1); %Guess for velocity (km/s)
PSI_Check(1,1) = PSI_Ref; %Guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t, Traj_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,HalfPeriod(Xing),r_Ref,V_Check(1,1), ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Check(1,1),bank);

if SMA_Target(Xing,2) == 1 %Find: Apogee radial position (km)
    [r_Check(1,1),ApogFlag] = max(Traj_States(:,1));
elseif SMA_Target(Xing,2) == 2 %Find: Perigee radial position (km)
    [r_Check(1,1),PerigFlag] = min(Traj_States(:,1));
end

%Semi-major axis (km)
SMA_Check(1,1) = 0.5*(r_Ref + r_Check(1,1));

%Iteration error (s)
GuessError(1,1) = -((SMA_Check(1,1) - SMA_Ref)/ ...
    ((V_Check(1,1) - V_Decrement) - V_Check(1,1)));

%Updated velocity (km/s)
V_Check(2,1) = (V_Check(1,1) - V_Decrement) - ...
    ((SMA_Target(Xing,1) - SMA_Check(1,1))/GuessError);

%Updated heading angle (rad)
PSI_Check(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check(2,1)));

```

```

%Difference between calculated and target trajectory states
IterDiff_SMA(1,1) = abs(SMA_Target(Xing,1) - SMA_Check(1,1));
IterDiff_PSI(1,1) = abs(PSI_Check(2,1) - PSI_Check(1,1));

IterCount = 1; %Initializes iteration counter for Newton-Raphson loop

%% Newton-Raphson Iteration
for ii = 2:IterMax
    while abs(SMA_Target(Xing,1) - SMA_Check(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check(ii,1) - PSI_Check(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t, Traj_States] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,HalfPeriod(Xing),r_Ref,V_Check(ii,1), ...
            lon_Ref,lat_Ref,fpa_Ref,PSI_Check(ii,1),bank);

        if SMA_Target(Xing,2) == 1 %Find: Perigee radial position (km)
            [r_Check(ii,1),PerigFlag] = max(Traj_States(:,1));

        elseif SMA_Target(Xing,2) == 2 %Find: Apogee radial position (km)
            [r_Check(ii,1),ApogFlag] = min(Traj_States(:,1));
        end

        %Current iteration semi-major axis (km)
        SMA_Check(ii,1) = 0.5*(r_Ref + r_Check(ii,1));

        %Iteration error (sec)
        GuessError(ii,1) = -((SMA_Check(ii,1) - SMA_Check(ii-1,1))/ ...
            (V_Check(ii,1) - V_Check(ii-1,1)));

        %Updated velocity (km/s)
        V_Check(ii+1,1) = V_Check(ii,1) - ...
            ((SMA_Target(Xing,1) - SMA_Check(ii,1))/GuessError(ii,1));

        %Updated heading angle (rad)
        PSI_Check(ii+1,1) = PSI_Ref + ...
            asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterDiff_SMA(ii,1) = abs(SMA_Target(Xing,1) - SMA_Check(ii,1));
        IterDiff_PSI(ii,1) = abs(PSI_Check(ii,1) - PSI_Check(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount = IterCount + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

V_Maneuver = V_Check(ii,1); %Maneuver velocity for target SMA
dV_Maneuver = abs(V_Maneuver - V_Rel); %Maneuver delta-V (km/s)
PSI_Maneuver = PSI_Check(ii,1); %Maneuver heading angle (deg)

```

```

%Trajectory simulation for skip maneuver
[Skip_t, Skip_States] = Maneuver_MainFunction(Vehicle_Choice,1,1,1,1,1, ...
      SMA_Skip(Xing,1)*(SMA_Skip(Xing,3)),r_Ref, ...
      V_Maneuver,lon_Ref,lat_Ref,fpa_Ref,PSI_Maneuver,bank);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Propagation of Re-Circularized Orbit
Time_Max = 5000;           %Maximum simulation time (s)
ecc       = 0;             %Orbit eccentricity
r_Prop    = Skip_States(end,1); %Orbit radial position (km)
h_Prop    = r_Prop - RE;    %Orbit altitude (km)
lon_Prop  = Skip_States(end,3); %Initial longitude (rad)
lat_Prop  = Skip_States(end,4); %Initial geodetic latitude (rad)
fpa_Prop  = 0;             %Flight-path angle (rad)
PSI_Prop  = PSI_Ref;       %Heading angle (rad)
bank      = 0;             %Bank angle (deg)

%Re-circularized orbit parameters
SMA_Prop   = 0.5*(r_Prop + r_Prop);           %Semi-major axis (km)
Period_Prop = (2*pi)*sqrt((SMA_Prop^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_RelProp, PSI_RelProp] = RelativeStates(mass,S,Cd,Cl,h_Prop,lat_Prop, ...
      fpa_Prop,PSI_Prop,bank);

SMA_TargetProp = SMA_Prop; %Target semi-major axis for iteration (km)
V_Decrement    = 1 - 0.9999; %Decrement value for velocity (km/s)
V_CheckProp(1,1) = V_RelProp; %Initial guess for velocity (km/s)
PSI_CheckProp(1,1) = PSI_Prop; %Initial guess for heading angle (rad)
IterMax        = 50; %Maximum number of iterations

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
      1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckProp(1,1), ...
      lon_Prop,lat_Prop,fpa_Prop,PSI_CheckProp(1,1),bank);
[r_CheckProp(1,1),ApogFlag] = max(Traj_StatesP(:,1));

%Semi-major axis (km)
SMA_CheckProp(1,1) = 0.5*(r_Prop + r_CheckProp(1,1));

%Iteration error (s)
GuessError_Prop(1,1) = -((SMA_CheckProp(1,1) - SMA_Prop)/ ...
      ((V_CheckProp(1,1) - V_Decrement) - V_CheckProp(1,1)));

%Updated velocity (km/s)
V_CheckProp(2,1) = (V_CheckProp(1,1) - V_Decrement) - ...
      ((SMA_TargetProp - SMA_CheckProp(1,1))/GuessError_Prop(1,1));

%Updated heading angle (rad)
PSI_CheckProp(2,1) = PSI_Prop + ...
      asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckProp(2,1)));

```



```

%Difference between calculated and target trajectory states
IterDiff_SMA_Prop(1,1) = abs(SMA_TargetProp - SMA_CheckProp(1,1));
IterDiff_PSI_Prop(1,1) = abs(PSI_CheckProp(2,1) - PSI_CheckProp(1,1));
IterCount_Prop = 1; %Initializes iteration counter for Newton-Raphson loop

%% Newton-Raphson Iteration
for ii = 2:IterMax
    while abs(SMA_TargetProp - SMA_CheckProp(ii-1,1)) > 1E-10 && ...
        abs(PSI_CheckProp(ii,1) - PSI_CheckProp(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_tP, Traj_StatesP] = Maneuver_MainFunction(Vehicle_Choice, ...
            1,1,1,1,1,0.5*Period_Prop,r_Prop,V_CheckProp(ii,1), ...
            lon_Prop,lat_Prop,fpa_Prop,PSI_CheckProp(ii,1),bank);

        [r_CheckProp(ii,1),ApogFlag] = max(Traj_StatesP(:,1));

        %Current iteration semi-major axis (km)
        SMA_CheckProp(ii,1) = 0.5*(r_Prop + r_CheckProp(ii,1));

        %Iteration error (sec)
        GuessError_Prop(ii,1) = -((SMA_CheckProp(ii,1) - ...
            SMA_CheckProp(ii-1,1))/ ...
            (V_CheckProp(ii,1) - V_CheckProp(ii-1,1)));

        %Updated velocity (km/s)
        V_CheckProp(ii+1,1) = V_CheckProp(ii,1) - ...
            ((SMA_TargetProp - SMA_CheckProp(ii,1))/GuessError_Prop(ii,1));

        %Updated heading angle (rad)
        PSI_CheckProp(ii+1,1) = PSI_Prop + ...
            asin((2*pi*(r_Prop)*sin(PSI_Prop))/(86400*V_CheckProp(ii+1,1)));

        %Difference between calculated and target trajectory states
        IterDiff_SMA_Prop(ii,1) = abs(SMA_TargetProp - SMA_CheckProp(ii,1));
        IterDiff_PSI_Prop(ii,1) = abs(PSI_CheckProp(ii,1) - ...
            PSI_CheckProp(ii-1,1));

        ii = ii + 1; %Update to row-index counter
        IterCount_Prop = IterCount_Prop + 1; %Update to iteration counter
    end
    break %Breaks row-index loop once tolerance is fulfilled
end

%Trajectory simulation for re-circularized orbit
[Orbit_t, Orbit_States] = Maneuver_MainFunction(Vehicle_Choice, ...
    1,1,1,1,1,Time_Max,r_Prop,V_CheckProp(ii), ...
    lon_Prop,lat_Prop,fpa_Prop,PSI_CheckProp(ii),bank);

%Concatenation of maneuver and orbit propagation time vectors
PropOrb_t = [Skip_t ; Skip_t(end) + Orbit_t(2:end)];

```

```

%Concatenation of maneuver and orbit propagation states
PropOrb_States = [Skip_States(:,1:6) ; Orbit_States(2:end,1:6)];
PropOrb_h       = PropOrb_States(:,1) - RE;      %Altitude (km)
PropOrb_Lon_deg = rem((rad2deg(PropOrb_States(:,3)) ...
                    + 180),360) - 180;          %Longitude (rad)
PropOrb_Lat_deg = rad2deg(PropOrb_States(:,4)); %Geocentric latitude (rad)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Propagated Trajectory Crossings of Target Coordinates
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lon_deg)
    mm = mm + 1;
    if abs(PropOrb_Lon_deg(ii) - Lon_Target) < 20
        LonTGT_Crossing(mm,1) = PropOrb_t(ii);
        LonTGT_Crossing(mm,2) = PropOrb_h(ii);
        LonTGT_Crossing(mm,3) = PropOrb_Lat_deg(ii);
        LonTGT_Crossing(mm,4) = PropOrb_Lon_deg(ii);
    else
        LonTGT_Crossing(mm,1:4) = 0;
    end
end

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lat_deg)
    mm = mm + 1;
    if abs(PropOrb_Lat_deg(ii) - Lat_Target) < 20
        LatTGT_Crossing(mm,1) = PropOrb_t(ii);
        LatTGT_Crossing(mm,2) = PropOrb_h(ii);
        LatTGT_Crossing(mm,3) = PropOrb_Lon_deg(ii);
        LatTGT_Crossing(mm,4) = PropOrb_Lat_deg(ii);
    else
        LatTGT_Crossing(mm,1:4) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Crossings
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(LonTGT_Crossing)
    if LonTGT_Crossing(ii) ~= 0
        mm = mm + 1;
        FlagVector_Lon(mm,1) = ii;
        WithinIdent_Lon(mm,:) = LonTGT_Crossing(ii,:);
    end
end
FlagVector_Lon = [FlagVector_Lon;0];

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(LatTGT_Crossing)
    if LatTGT_Crossing(ii) ~= 0
        mm = mm + 1;

```

```

        FlagVector_Lat(mm,1) = ii;
        WithinIdent_Lat(mm,:) = LatTGT_Crossing(ii,:);
    end
end
FlagVector_Lat = [FlagVector_Lat;0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Crossings
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Lon)-1
    if abs((FlagVector_Lon(ii+1) - FlagVector_Lon(ii))) > 1
        mm = mm + 1;
        LonTGT_Jump(mm,1) = ii;
    end
end
LonTGT_Jump = [0;LonTGT_Jump];

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector_Lat)-1
    if abs((FlagVector_Lat(ii+1) - FlagVector_Lat(ii))) > 1
        mm = mm + 1;
        LatTGT_Jump(mm,1) = ii;
    end
end
LatTGT_Jump = [0;LatTGT_Jump];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Crossing Trajectories
%Longitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(LonTGT_Jump)
    mm = mm + 1;
    LonTGT_Interp(mm,:) = ...
        interp1(LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
            FlagVector_Lon(LonTGT_Jump(ii)),4), ...
            LonTGT_Crossing(FlagVector_Lon(LonTGT_Jump(ii-1)+1): ...
            FlagVector_Lon(LonTGT_Jump(ii)),1:3), ...
            Lon_Target,'spline'); %Cubic spline interpolation
end

%Latitude crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(LatTGT_Jump)
    mm = mm + 1;
    LatTGT_Interp(mm,:) = ...
        interp1(LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
            FlagVector_Lat(LatTGT_Jump(ii)),4), ...
            LatTGT_Crossing(FlagVector_Lat(LatTGT_Jump(ii-1)+1): ...
            FlagVector_Lat(LatTGT_Jump(ii)),1:3), ...
            Lat_Target,'spline'); %Cubic spline interpolation
end

```

```

%Removal of negative perturbed periods
LatTGT_Interp(any(LatTGT_Interp(:,1)<0,2),:) = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Minimum Target Miss Distance
%Target miss distance for both spherical and oblate planetary models
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lon_deg)
    mm = mm + 1;
    SphereDist_Lon = CoordDist(Lon_Target,Lon_Target, ...
                               Lat_Target,LonTGT_Interp(:,3),1);
end

%Longitudinal target miss distance (km)
[MinDistance_Lon,MinFlag_Lon] = min(SphereDist_Lon(:,1));

mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(PropOrb_Lat_deg)
    mm = mm + 1;
    SphereDist_Lat = CoordDist(Lon_Target,LatTGT_Interp(:,3), ...
                               Lat_Target,Lat_Target,1);
end

%Latitudinal target miss distance (km)
[MinDistance_Lat,MinFlag_Lat] = min(SphereDist_Lat(:,1));

MinDist_Vec = [MinDistance_Lon, MinDistance_Lat]; %Miss distance vector
MinFlag_Vec = [MinFlag_Lon,      MinFlag_Lat];    %Minimum flag vector

[MinDistance, MinIndex] = min(MinDist_Vec); %Minimum miss distance
MinFlag         = MinFlag_Vec(MinIndex);    %Flag for minimum miss distance

%Determination of interpolated data set associated with min. miss distance
if      MinIndex == 1
    MinInterp = LonTGT_Interp; %Interpolated data for longitude crossing
elseif  MinIndex == 2
    MinInterp = LatTGT_Interp; %Interpolated data for latitude crossing
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Total Skip Maneuver Delta-V
V_EndSkip = Skip_States(end,2); %Velocity where fpa = 0 (km/s)
dV_ReCirc = abs(V_EndSkip - V_RelProp); %Re-circularization delta-V (km/s)

%Total delta-V for skip maneuver (km/s)
dV_SkipTotal = dV_Maneuver + dV_ReCirc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Arrival and Trajectory Parameters
%Time-of-arrival at target (hr)
TimeArrival = (MinInterp(MinFlag,1))*(1/60)*(1/60);

%Altitude-of-arrival at target (km)
AltArrival = MinInterp(MinFlag,2);

```

```

PertOrbNum = SMA_Skip(Xing,3); %Number of perturbed orbits
TypeFlag   = SMA_Target(Xing,2); %Type of maneuver flag

h_Apog      = r_Apog(Xing) - RE; %Skip apogee altitude (km)
h_Perig     = r_Perig(Xing) - RE; %Skip perigee altitude (km)
SkipEcc     = ((r_Apog(Xing) - r_Perig(Xing))/ ...
               (r_Apog(Xing) + r_Perig(Xing))); %Eccentricity

%Payload imager field-of-view (FOV) and resolution during over-flight
%Visible spectrum imager
[FOV_m2_Vis, FOV_km2_Vis, Resolution_Vis] = ...
    PayloadImager(AltArrival*(1.0E3),1.15,2.70,1.0E-6);
%Infrared spectrum imager
[FOV_m2_IR, FOV_km2_IR, Resolution_IR] = ...
    PayloadImager(AltArrival*(1.0E3),1.15,2.70,11.0E-6);

%Over-flight parameter matrix
Trajectory_Analysis = [TypeFlag, TimeArrival, AltArrival, ...
    SMA_Skip(Xing,4), PertOrbNum, h_Apog, h_Perig, ...
    SkipEcc, Resolution_Vis, dV_SkipTotal, ...
    MinDistance, dV_Maneuver, dV_ReCirc, Resolution_IR];

%Prints notification of maneuver simulation completion to command window
fprintf('Simulation Run: %d \n',Xing);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Re-Defines Data to Reflect Jumps in Data between 180 and -180 deg
%Reference orbit
[Lon_RefOrb, Lat_RefOrb, LonSplit_RefOrb, LatSplit_RefOrb] = ...
    CoordinateJump(RefOrb_States);
%Maneuver orbit
[Lon_Skip, Lat_Skip, LonSplit_Skip, LatSplit_Skip] = ...
    CoordinateJump(Skip_States);
%Propagated re-circularized orbit
[Lon_PropOrb, Lat_PropOrb, LonSplit_PropOrb, LatSplit_PropOrb] = ...
    CoordinateJump(PropOrb_States);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Command Window Printing
fprintf('Maneuver Velocity: %f km/s \n', V_Maneuver);
fprintf('Maneuver Heading Angle: %f deg \n', rad2deg(PSI_Maneuver));
fprintf('Number of Iterations: %d \n', IterCount);
fprintf('Minimum Miss Distance: %f km \n', MinDistance);
fprintf('Time-of-Arrival: %f hr \n', TimeArrival);
fprintf('Maneuver Delta-V: %f km/s \n', dV_Maneuver);
fprintf('Total Delta-V: %f km/s \n', dV_SkipTotal);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Commands
%Conversion of time units for plotting
[Skip_Time, time_string] = TimeUtility(Skip_t,2);
[PropOrb_Time, time_string] = TimeUtility(PropOrb_t,2);

```

```

%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,1); box on; grid off;
hold on; cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);
xlim([-180 180]); ylim([-90 90]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90]; p2_Lon = [Lon_Target, 90];

%Target latitude, longitude lines
hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)], 'r:');
hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)], 'r:');

hold on; %Plate Carree world map projection
landareas = shaperead('landareas.shp', 'UseGeoCoords',true);
geoshow(landareas, 'FaceColor',[1 1 .5], 'EdgeColor',[.6 .6 .6]);

%Target location
hold on; plot(Lon_Target,Lat_Target, 'o', 'MarkerEdgeColor','r', ...
              'MarkerFaceColor','r', 'MarkerSize',5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geocentric Latitude (deg) v. Longitude (deg)
subplot(2,2,2); box on; grid off;
hold on; cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);
% xlim([-180 180]); ylim([-90 90]);
xlim([-180 180]);
ylim([floor(Lat_Target)-5, ceil(Lat_Target)+5]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90]; p2_Lon = [Lon_Target, 90];

%Trajectory crossings of target latitude
hold on;
plot(CrossingIdent(:,3),CrossingIdent(:,4), 'ko');
hold on;
plot(CrossInterp(:,3),Lat_Target, 'gs', 'LineWidth',2);

%Target latitude, longitude lines
hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)], 'r--');
hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)], 'r--');

%Target location
hold on; plot(Lon_Target,Lat_Target, 'o', 'MarkerEdgeColor','r', ...
              'MarkerFaceColor','r', 'MarkerSize',5);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
subplot(2,2,3); box on; grid off;
hold on;
h_Ref = cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);
hold on;
h_Prop = cellfun(@plot,LonSplit_PropOrb,LatSplit_PropOrb);

set(h_Ref, 'LineStyle','-','Color','b');
set(h_Prop, 'LineStyle','--','Color','r');

xlim([-180 180]); ylim([-90 90]);
% xlim([floor(Lon_Target)-10, ceil(Lon_Target)+10]);
% ylim([floor(Lat_Target)-10, ceil(Lat_Target)+10]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

% legend('Reference Orbit','Perturbed Orbit','Location','NorthEast');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90]; p2_Lon = [Lon_Target, 90];

%Target latitude, longitude lines
hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)], 'r:');
hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)], 'r:');

hold on; %Plate Carree world map projection
landareas = shaperead('landareas.shp','UseGeoCoords',true);
geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

%Target location
hold on; plot(Lon_Target,Lat_Target,'o','MarkerEdgeColor','r', ...
             'MarkerFaceColor','r','MarkerSize',5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geodetic Altitude (km) vs. Time (sec)
[PropOrb_t, time_string] = TimeUtility(PropOrb_t,2); %Time unit conversion
subplot(2,2,4); box on; grid on;

plot(PropOrb_t,PropOrb_h,'b');
xlabel(['Time, ', time_string]);
ylabel('Geodetic Altitude, km');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

RefOrb_Targeting.m

```
clear all; clc; close all;

global MU RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape

WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Target Selection and Targeting Loop Initialization
Vehicle_Choice = 1;
Target_Choice = 12;

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver; Lon_Target = Lon_Denver; dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow; Lon_Target = Lon_Glasgow; dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny; Lon_Target = Lon_Grozny; dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow; Lon_Target = Lon_Moscow; dLat = 3;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti; Lon_Target = Lon_Ponti; dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong; Lon_Target = Lon_Pyong; dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk; Lon_Target = Lon_Reyk; dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran; Lon_Target = Lon_Tehran; dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo; Lon_Target = Lon_Tokyo; dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil; Lon_Target = Lon_Brasil; dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos; Lon_Target = Lon_Buenos; dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra; Lon_Target = Lon_Canberra; dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape; Lon_Target = Lon_Cape; dLat = 4;
end
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Model
if Vehicle_Choice == 9      %VEHICLE SELECTION OVERRIDE
    mass = 2000;           %Mass (kg)
    S_m2 = 18.5;           %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = 0.5;           %Drag coefficient
    Cl    = 3.0;           %Lift coefficient
else
    [Vehicle] = VehicleSpecs(Vehicle_Choice);
    mass = Vehicle.mass;   %Mass (kg)
    S_m2 = Vehicle.S_m2;   %Planform area (m^2)
    S     = S_m2/(1000^2); %Planform area (km^2)
    Cd    = Vehicle.Cd;    %Drag coefficient
    Cl    = Vehicle.Cl;    %Lift coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PSI_Ref0      = 60;        %Initial reference orbit heading angle (deg)
PSI_Ref       = PSI_Ref0; %Initial estimate for heading angle (deg)
MissDistance  = 9999; %Initializes 'MissDistance' variable for targeting loop
WhileCount    = 0;        %Initializes 'while'-loop iteration counter

while MissDistance > 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Reference Orbit Conditions
Time_Max = 1;      %Maximum simulation time (days)
ecc_Ref  = 0;      %Orbit eccentricity
h_Ref    = 1000;   %Orbit geodetic altitude (km)
lon_Ref  = 0;      %Initial longitude (deg)
lat_Ref  = 0;      %Initial geodetic latitude (deg)
fpa_Ref  = 0;      %Flight-path angle (deg)
bank     = 0;      %Bank angle (deg)

%Converts and overwrites initial angle variables
lon_Ref = deg2rad(lon_Ref); lat_Ref = deg2rad(lat_Ref);
fpa_Ref = deg2rad(fpa_Ref); PSI_Ref = deg2rad(PSI_Ref);

%Reference orbit parameters
r_Ref    = h_Ref + RE; %Radial position (km)
SMA_Ref  = 0.5*(r_Ref + r_Ref); %Semi-major axis (km)
RefPeriod = (2*pi)*sqrt((SMA_Ref^3)/MU); %Orbit period (sec)

%Velocity relative to rotating frame (rotating planet)
[V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_Ref,lat_Ref, ...
                                fpa_Ref,PSI_Ref,bank);

%Conversion of time units from days to seconds
Time_Max = Time_Max*(24)*(60)*(60);
SMA_Target0 = SMA_Ref; %Target semi-major axis (km)
V_Decrement = 1 - 0.9999; %Decrement value for velocity (km/s)
V_Check0(1,1) = V_Rel; %Initial guess for velocity (km/s)
PSI_Check0(1,1) = PSI_Ref; %Initial guess for heading angle (rad)
IterMax = 50; %Maximum number of iterations

```

```

%% First Iteration
%Trajectory simulation [0:t:HalfPeriod]
[Traj_t0, Traj_States0] = Maneuver_MainFunction(1,1,1,1,1,1, ...
    0.5*RefPeriod,r_Ref,V_Check0(1,1), ...
    lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(1,1),bank);

[r_Check0(1,1),ApogFlag] = max(Traj_States0(:,1));

%Semi-major axis (km)
SMA_Check0(1,1) = 0.5*(r_Ref + r_Check0(1,1));

%Iteration error (s)
GuessError0(1,1) = -((SMA_Check0(1,1) - SMA_Ref)/ ...
    ((V_Check0(1,1) - V_Decrement) - V_Check0(1,1)));

%Updated velocity (km/s)
V_Check0(2,1) = (V_Check0(1,1) - V_Decrement) - ...
    ((SMA_Target0 - SMA_Check0(1,1))/GuessError0(1,1));

%Updated heading angle (rad)
PSI_Check0(2,1) = PSI_Ref + ...
    asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check0(2,1)));

%Difference between calculated and target trajectory states
IterativeDiff_SMA0(1,1) = abs(SMA_Target0 - SMA_Check0(1,1));
IterativeDiff_PSI0(1,1) = abs(PSI_Check0(2,1) - PSI_Check0(1,1));

IterCount = 1; %Initializes iteration counter for Newton-Raphson loop

%% Newton-Raphson Iteration
for ii = 2:IterMax
    while abs(SMA_Target0 - SMA_Check0(ii-1,1)) > 1E-10 && ...
        abs(PSI_Check0(ii,1) - PSI_Check0(ii-1,1)) > 1E-10

        %Trajectory simulation [0:t:HalfPeriod]
        [Traj_t0, Traj_States0] = Maneuver_MainFunction(1,1,1,1,1,1, ...
            0.5*RefPeriod,r_Ref,V_Check0(ii,1), ...
            lon_Ref,lat_Ref,fpa_Ref,PSI_Check0(ii,1),bank);

        [r_Check0(ii,1),ApogFlag] = max(Traj_States0(:,1));

        %Current iteration semi-major axis (km)
        SMA_Check0(ii,1) = 0.5*(r_Ref + r_Check0(ii,1));

        %Iteration error (sec)
        GuessError0(ii,1) = -((SMA_Check0(ii,1) - SMA_Check0(ii-1,1))/ ...
            (V_Check0(ii,1) - V_Check0(ii-1,1)));

        %Updated velocity (km/s)
        V_Check0(ii+1,1) = V_Check0(ii,1) - ...
            ((SMA_Target0 - SMA_Check0(ii,1))/GuessError0(ii,1));
    end
end

```

```

    %Updated heading angle (rad)
    PSI_Check0(ii+1,1) = PSI_Ref + ...
        asin((2*pi*(r_Ref)*sin(PSI_Ref))/(86400*V_Check0(ii+1,1)));

    %Difference between calculated and target trajectory states
    IterativeDiff_SMA0(ii,1) = abs(SMA_Target0 - SMA_Check0(ii,1));
    IterativeDiff_PSI0(ii,1) = abs(PSI_Check0(ii,1) - ...
        PSI_Check0(ii-1,1));

    ii = ii + 1; %Update to row-index counter
    IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

V_Rel0 = V_Check0(ii); %Velocity (km/s)
PSI_Rel0 = PSI_Check0(ii); %Heading angle (rad)

%Trajectory simulation for reference orbit
[RefOrb_t,RefOrb_States] = Maneuver_MainFunction(1,1,1,1,1,1,Time_Max, ...
    r_Ref,V_Rel0,lon_Ref,lat_Ref,fpa_Ref,PSI_Rel0,bank);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Manipulation
r_Data = RefOrb_States(:,1); %Radial position (km)
h_Data = r_Data - RE; %Altitude (km)
Lon_Data = RefOrb_States(:,3); %Longitude (rad)
Lat_Data = RefOrb_States(:,4); %Geocentric latitude (rad)

%Transforms longitude from (0 <= lon < 360) to (-180 < lon <= 180)
Lon_Data = rem((rad2deg(Lon_Data) + 180),360) - 180;

%Converts geodetic latitude from radians to degrees
Lat_Data = rad2deg(Lat_Data);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Trajectory Crossings of Target Longitude
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Lon_Data)
    mm = mm + 1;
    if abs(Lon_Data(ii) - Lon_Target) < 10
        LonCrossing(mm,1) = RefOrb_t(ii);
        LonCrossing(mm,2) = h_Data(ii);
        LonCrossing(mm,3) = Lat_Data(ii);
        LonCrossing(mm,4) = Lon_Data(ii);
    else
        LonCrossing(mm,1:4) = 0;
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Longitude Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(LonCrossing)
    if LonCrossing(ii) ~= 0
        mm = mm + 1;
        FlagVector(mm,1) = ii;
        WithinIdent(mm,:) = LonCrossing(ii,:);
    end
end
FlagVector = [FlagVector;0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Indices Corresponding to Jumps in Longitude Crossings
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(FlagVector)-1
    if abs((FlagVector(ii+1) - FlagVector(ii))) > 1
        mm = mm + 1;
        LonJump(mm,1) = ii;
    end
end
LonJump = [0;LonJump];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Interpolation of Longitude Crossing Trajectories
mm = 0; %Initializes vector concatenation counter at zero
for ii = 2:length(LonJump)
    mm = mm + 1;
    LonInterp(mm,:) = ...
        interp1(LonCrossing(FlagVector(LonJump(ii-1)+1): ...
            FlagVector(LonJump(ii)),4), ...
            LonCrossing(FlagVector(LonJump(ii-1)+1): ...
            FlagVector(LonJump(ii)),1:3), ...
            Lon_Target,'spline'); %Cubic spline interpolation
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Minimum Target Miss Distance
%Target miss distance for both spherical and oblate planetary models
mm = 0; %Initializes vector concatenation counter at zero
for ii = 1:length(Lat_Data)
    mm = mm + 1;
    SphereDist = ...
        CoordDist(Lon_Target,Lon_Target,Lat_Target,LonInterp(:,3),1);
end

%Target miss distance (km)
[MinDistance,MinFlag] = min(SphereDist(:,1));
MissDistance = MinDistance

%Time-of-arrival at target (hr)
TimeArrival = (LonInterp(MinFlag,1))*(1/60)*(1/60);

%Latitude-of-arrival at target (deg)
LatArrival = LonInterp(MinFlag,3);

```

```

if      LatArrival > Lat_Target
    if      MissDistance > 1000
        PSI_Ref = rad2deg(PSI_Ref) + 1.0;
    elseif MissDistance > 100 && MissDistance <= 1000
        PSI_Ref = rad2deg(PSI_Ref) + 0.5;
    elseif MissDistance > 20 && MissDistance <= 100
        PSI_Ref = rad2deg(PSI_Ref) + 0.1;
    elseif MissDistance > 3 && MissDistance <= 20
        PSI_Ref = rad2deg(PSI_Ref) + 0.01;
    elseif MissDistance <= 3
        PSI_Ref = rad2deg(PSI_Ref) + 0.001;
    end

elseif LatArrival < Lat_Target
    if      MissDistance > 1000
        PSI_Ref = rad2deg(PSI_Ref) - 1.0;
    elseif MissDistance > 100 && MissDistance <= 1000
        PSI_Ref = rad2deg(PSI_Ref) - 0.5;
    elseif MissDistance > 20 && MissDistance <= 100
        PSI_Ref = rad2deg(PSI_Ref) - 0.1;
    elseif MissDistance > 3 && MissDistance <= 20
        PSI_Ref = rad2deg(PSI_Ref) - 0.01;
    elseif MissDistance <= 3
        PSI_Ref = rad2deg(PSI_Ref) - 0.001;
    end
end

WhileCount = WhileCount + 1; %Update to 'while'-loop iteration counter

%Clearing of variables for targeting loop
clear LonCrossing; clear FlagVector; clear WithinIdent;
clear LonJump;      clear LonInterp;

end

%Payload imager field-of-view (FOV) and resolution during over-flight
%Visible spectrum imager
[FOV_m2_Vis,FOV_km2_Vis,Resolution_Vis] = ...
    PayloadImager(h_Ref*(1.0E3),1.15,2.70,1.0E-6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of Simple Plane Change Delta-V
%Initial velocity relative to rotating frame (rotating planet)
[V_Rel_Init,PSI_Rel_Init] = RelativeStates(mass,S,Cd,Cl,h_Ref,lat_Ref, ...
    fpa_Ref,deg2rad(PSI_Ref0),bank);

dV_Simple = InclinationChange(V_Rel_Init,fpa_Ref,abs(PSI_Ref0 - PSI_Ref));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Re-Defines Data to Reflect Jumps in Data between 180 and -180 deg
%Reference orbit
[Lon_RefOrb, Lat_RefOrb, LonSplit_RefOrb,LatSplit_RefOrb] = ...
    CoordinateJump(RefOrb_States);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Command Window Printing
fprintf('Minimum Miss Distance:           %f km   \n', MinDistance);
fprintf('Time-of-Arrival:                 %f hr   \n', TimeArrival);
fprintf('Simple Plane Change Delta-V:      %f km/s \n', dV_Simple);

return

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting Commands
%Geocentric Latitude (deg) v. Longitude (deg) with Plate Carree Projection
figure; box on; grid on;
hold on; cellfun(@plot,LonSplit_RefOrb, LatSplit_RefOrb);
xlim([-180 180]); ylim([-90 90]);
xlabel('Longitude, deg');
ylabel('Geocentric Latitude, deg');

%Coordinates (x,y) for target latitude, longitude end points
p1_Lat = [-180, Lat_Target]; p2_Lat = [180, Lat_Target];
p1_Lon = [Lon_Target, -90];  p2_Lon = [Lon_Target, 90];

%Target latitude, longitude lines
hold on; plot([p1_Lat(1),p2_Lat(1)],[p1_Lat(2),p2_Lat(2)],'r:');
hold on; plot([p1_Lon(1),p2_Lon(1)],[p1_Lon(2),p2_Lon(2)],'r:');

% hold on; %Plate Carree world map projection
% landareas = shaperead('landareas.shp','UseGeoCoords',true);
% geoshow(landareas,'FaceColor',[1 1 .5],'EdgeColor',[.6 .6 .6]);

%Target location
hold on; plot(Lon_Target,Lat_Target,'o','MarkerEdgeColor','r', ...
              'MarkerFaceColor','r','MarkerSize',5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

Trajectory_3DPlotting.m

```
global RE
global Lat_Denver Lon_Denver Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow Lon_Glasgow Lat_Grozny Lon_Grozny
global Lat_Moscow Lon_Moscow Lat_Ponti Lon_Ponti
global Lat_Pyong Lon_Pyong Lat_Reyk Lon_Reyk
global Lat_Tehran Lon_Tehran Lat_Tokyo Lon_Tokyo
global Lat_Brasil Lon_Brasil Lat_Buenos Lon_Buenos
global Lat_Canberra Lon_Canberra Lat_Cape Lon_Cape

WGS84Constants; %Loads global constants from external m-file
GroundTargets; %Loads ground target geographical coordinates (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Target Selection
Target_Choice = 5;

if Target_Choice == 1 %Denver, United States
    Lat_Target = Lat_Denver; Lon_Target = Lon_Denver; dLat = 3;
elseif Target_Choice == 2 %Gibraltar, United Kingdom
    Lat_Target = Lat_Gibraltar; Lon_Target = Lon_Gibraltar; dLat = 5;
elseif Target_Choice == 3 %Glasgow, Scotland
    Lat_Target = Lat_Glasgow; Lon_Target = Lon_Glasgow; dLat = 3;
elseif Target_Choice == 4 %Grozny, Chechnya
    Lat_Target = Lat_Grozny; Lon_Target = Lon_Grozny; dLat = 3;
elseif Target_Choice == 5 %Moscow, Russia
    Lat_Target = Lat_Moscow; Lon_Target = Lon_Moscow; dLat = 3;
elseif Target_Choice == 6 %Pontianak, Indonesia
    Lat_Target = Lat_Ponti; Lon_Target = Lon_Ponti; dLat = 7;
elseif Target_Choice == 7 %Pyongyang, North Korea
    Lat_Target = Lat_Pyong; Lon_Target = Lon_Pyong; dLat = 5;
elseif Target_Choice == 8 %Reykjavik, Iceland
    Lat_Target = Lat_Reyk; Lon_Target = Lon_Reyk; dLat = 3;
elseif Target_Choice == 9 %Tehran, Iran
    Lat_Target = Lat_Tehran; Lon_Target = Lon_Tehran; dLat = 5;
elseif Target_Choice == 10 %Tokyo, Japan
    Lat_Target = Lat_Tokyo; Lon_Target = Lon_Tokyo; dLat = 4;
elseif Target_Choice == 11 %Brasilia, Brazil
    Lat_Target = Lat_Brasil; Lon_Target = Lon_Brasil; dLat = 4;
elseif Target_Choice == 12 %Buenos Aires, Argentina
    Lat_Target = Lat_Buenos; Lon_Target = Lon_Buenos; dLat = 4;
elseif Target_Choice == 13 %Canberra, Australia
    Lat_Target = Lat_Canberra; Lon_Target = Lon_Canberra; dLat = 4;
elseif Target_Choice == 14 %Cape Town, South Africa
    Lat_Target = Lat_Cape; Lon_Target = Lon_Cape; dLat = 4;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 3-D Globe Model
figure; clf reset;
Earth = referenceSphere('earth','km');
Earth.Radius = RE;
```

```

ax = axesm('globe','Geoid',Earth,'Grid','off', ...
           'GLineWidth',1,'GLineStyle','-',' ...
           'Gcolor',[0.9 0.9 0.1],'Galtitude',100);

set(ax,'Position',[0 0 1 1]);
axis equal off; view(3); load topo;
geoshow(topo,topolegend,'DisplayType','texturemap');
demcmap(topo);
land = shaperead('landareas','UseGeoCoords',true);
plotm([land.Lat],[land.Lon],'Color','black'); hold on;
rivers = shaperead('worldrivers','UseGeoCoords',true);
plotm([rivers.Lat],[rivers.Lon],'Color','blue'); hold on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 3-D Ground Track Trajectory
%Initial maneuver and propagated trajectory
VecEnd = 341;
LatProp_Data = PropOrb_States(1:VecEnd,4);
LonProp_Data = PropOrb_States(1:VecEnd,3);
AltProp_Data = PropOrb_h(1:VecEnd,1);

% plotm(rad2deg(Lat_Data),rad2deg(Lon_Data),'k-','MarkerSize',2); hold on;
X_Prop = (RE + 1.*AltProp_Data).*sin((pi/2) -
LatProp_Data).*cos(LonProp_Data);
Y_Prop = (RE + 1.*AltProp_Data).*sin((pi/2) -
LatProp_Data).*sin(LonProp_Data);
Z_Prop = (RE + 1.*AltProp_Data).*cos((pi/2) - LatProp_Data);
plot3(X_Prop,Y_Prop,Z_Prop,'y-','LineWidth',1.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Molniya 3-42 Orbit Trajectory
load PropOrb_States_342;
load PropOrb_h_342;
load PropOrb_t_342;

VecEnd2 = 131;
LatProp_Data342 = PropOrb_States_342(1:VecEnd2,4);
LonProp_Data342 = PropOrb_States_342(1:VecEnd2,3);
AltProp_Data342 = PropOrb_h_342(1:VecEnd2,1);

X_Prop342 = (RE + 1.*AltProp_Data342).*sin((pi/2) -
LatProp_Data342).*cos(LonProp_Data342);
Y_Prop342 = (RE + 1.*AltProp_Data342).*sin((pi/2) -
LatProp_Data342).*sin(LonProp_Data342);
Z_Prop342 = (RE + 1.*AltProp_Data342).*cos((pi/2) - LatProp_Data342);
plot3(X_Prop342,Y_Prop342,Z_Prop342,'g-','LineWidth',1.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Maneuver trajectory
LatSkip_Data = Skip_States(:,4);
LonSkip_Data = Skip_States(:,3);
AltSkip_Data = Skip_States(:,1) - RE;

```



```

X_Skip      = (RE + 1.*AltSkip_Data).*sin((pi/2) -
LatSkip_Data).*cos(LonSkip_Data);
Y_Skip      = (RE + 1.*AltSkip_Data).*sin((pi/2) -
LatSkip_Data).*sin(LonSkip_Data);
Z_Skip      = (RE + 1.*AltSkip_Data).*cos((pi/2) - LatSkip_Data);
plot3(X_Skip,Y_Skip,Z_Skip,'r-','LineWidth',1.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','k'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

Appendix G. MATLAB® Code for Support Functions and Utilities

Table G.1. m-File Classification for Support Functions and Utilities

Filename	File Type	Description
COE2RV	Function	Converts COEs to \vec{r}, \vec{V}
CoordDist	Function	Calculates geodesies
CoordinateJump	Function	Shifts data to $-180^\circ < \theta < 180^\circ$
DescentDeltaV	Function	Calculates impulse to alter γ
Eccentricity	Function	Calculates orbit eccentricity
EntryDecel	Function	Calculates re-entry deceleration
FirstSkip	Function	Calculates single skip parameters
Geocentric2Geodetic	Function	Calculates geodetic coordinates
Geodetic2Geocentric	Function	Calculates geocentric coordinates
GroundTargets	Function	Coordinates of sample targets
HeatFluxModel	Function	Convective and radiative models
InclinationChange	Function	Simple plane change
KeplerAnomalies	Function	Calculates eccentric/true anomalies
OrbitVelocity	Function	For circular/elliptical orbits
PayloadImager	Function	Calculates FOV and resolution
RelativeStates	Function	States relative to rotating frame
RelativeStates_Entry	Function	States relative to rotating frame
ROT	Function	Rotation matrices
SingleSkip_Maneuver	Function	Calculates single skip maneuver
SMARadii	Function	Calculates apogee and perigee
TimeUtility	Function	User-defined time scale for plotting
TLE2RV	Function	Converts TLE to \vec{r}, \vec{V}

COE2RV.m

```
function [r,V] = COE2RV(MU,a,ecc,inc,raan,omega,nu,Angle_Choice)

%Conversion of classical elements from degrees to radians
if Angle_Choice == 1 %Angle conversion NOT required
    inc    = inc;    %Inclination (rad)
    raan   = raan;   %Right ascension of the ascending node (rad)
    omega  = omega;  %Argument of perigee (rad)
    nu     = nu;     %True anomaly (rad)
elseif Angle_Choice == 2 %Angle conversion required
    inc    = deg2rad(inc);
    raan   = deg2rad(raan);
    omega  = deg2rad(omega);
    nu     = deg2rad(nu);
end

%Semi-parameter (km)
p        = a*(1 - (ecc^2));

%Radial position vector in PQW-frame
Rpqw = [((p*cos(nu))/(1 + ecc*cos(nu))); ...
        ((p*sin(nu))/(1 + ecc*cos(nu))); 0];

%Velocity vector in PQW-frame
Vpqw = [-(sqrt(MU/p))*sin(nu); ...
        (sqrt(MU/p))*(ecc + cos(nu)); 0];

%Rotation from PQW-frame to IJK-frame
Rijk = ROT(3,-raan,1)*ROT(1,-inc,1)*ROT(3,-omega,1)*Rpqw;
Vijk = ROT(3,-raan,1)*ROT(1,-inc,1)*ROT(3,-omega,1)*Vpqw;

%Re-assignment of vector variable names
r      = Rijk; %(km)
V      = Vijk; %(km/s)
```

CoordDist.m

```
function [GeoDist] = CoordDist(Lon1,Lon2,Lat1,Lat2,Model_Choice)

global RE FlatE

WGS84Constants; %Loads global constants FlatE from external m-FlatE file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Conversion from degrees to radians
%Note: Longitude range = [0:theta:360] or [0:theta:2*pi]
Lon1 = deg2rad(Lon1); Lat1 = deg2rad(Lat1);
Lon2 = deg2rad(Lon2); Lat2 = deg2rad(Lat2);

a2 = RE^2;           %Square of semi-major axis (km)
b  = RE*(1-FlatE);   %Semi-minor axis (km)
b2 = b^2;           %Square of semi-minor axis (km)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Model_Choice == 1
% Spherical Planet Model: Distance between Two Coordinates (Great Circle)
GeoDist = RE*acos(sin(Lat1)*sin(Lat2) + ...
    cos(Lat1)*cos(Lat2)*cos(Lon1 - Lon2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif Model_Choice == 2
% Oblate Planet Model: Distance between Two Coordinates (Vincenty's Method)

%Reduced latitude (rad)
U1 = atan((1-FlatE)*tan(Lat1));
U2 = atan((1-FlatE)*tan(Lat2));

%Longitude difference (rad)
L = Lon2 - Lon1;
Lam = L; %Initial longitude diff. guess on auxiliary sphere (rad)
Old_Lam = 0; %Initialization of preceding iteration variable (rad)
IterCount = 0; %Initialization of iteration counter
IterMax = 50; %Maximum number of iterations

for ii = 1:IterMax
    while abs(Lam - Old_Lam) > 1E-12
        Old_Lam = Lam;

        SIN_Sigma = sqrt(((cos(U2)*sin(Old_Lam))^2) + ...
            ((cos(U1)*sin(U2) - sin(U1)*cos(U2)*cos(Old_Lam))^2));
        COS_Sigma = sin(U1)*sin(U2) + cos(U1)*cos(U2)*cos(Old_Lam);
        AngularDist = atan2(SIN_Sigma,COS_Sigma);

        SIN_Alpha = ((cos(U1)*cos(U2)*sin(Old_Lam))/sin(AngularDist));
        COS_Alpha2 = 1 - (SIN_Alpha^2);
        COS_2Sigma_m = cos(AngularDist) - ((2*sin(U1)*sin(U2))/COS_Alpha2);
        C = (FlatE/16)*COS_Alpha2*(4 + FlatE*(4 - 3*COS_Alpha2));
```

```

Lam = L + (1-C)*FlatE*SIN_Alpha*(AngularDist + ...
        C*SIN_Sigma*(COS_2Sigma_m + ...
        C*COS_Sigma*(-1 + 2*(COS_2Sigma_m^2))));

%Difference between current and preceding longitude difference
IterativeDiff_Lam(ii,1) = abs(Lam - Old_Lam);

ii = ii + 1; %Update to row-index counter
IterCount = IterCount + 1; %Update to iteration counter
end
break %Breaks row-index loop once tolerance is fulfilled
end

u2 = ((a2 - b2)/b2)*COS_Alpha2;
A = 1 + (u2/16384)*(4096 + u2*(-768 + u2*(320 - 175*u2)));

B = (u2/1024)*(256 + u2*(-128 + u2*(74 - 47*u2)));
dAngularDist = B*SIN_Sigma*(COS_2Sigma_m + ...
        (1/4)*B*(-1 + 2*(COS_2Sigma_m^2) - ...
        (1/6)*B*COS_2Sigma_m*(-3 + 4*(SIN_Sigma^2))* ...
        (-3 + 4*(COS_2Sigma_m^2))));
GeoDist = abs(b*A*(AngularDist - dAngularDist));

end

```

CoordinateJump.m

```

function [Lon_Data,Lat_Data,Lon_Split,Lat_Split] = ...
        CoordinateJump(traj_states)

lon = traj_states(:,3); %Longitude (rad)
lat = traj_states(:,4); %Latitude (rad)

%Transforms longitude from (0 <= lon < 360) to (-180 < lon <= 180)
Lon_Data = rem((rad2deg(lon')+ 180),360) - 180;

%Converts geocentric latitude from radians to degrees
Lat_Data = rad2deg(lat');

%Re-defines data to reflect jumps in data between 180 and -180 deg
Lon_Jumps = [0 find(abs(diff(Lon_Data))> 90) length(Lon_Data)];
Lon_Split = mat2cell(Lon_Data,1,diff(Lon_Jumps));
Lat_Split = mat2cell(Lat_Data,1,diff(Lon_Jumps));

```

DescentDeltaV.m

```
function [dV1,V1] = DescentDeltaV(h_orbit,h_atm,fpa0_deg)

global MU RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions
r0      = h_orbit + RE;      %Radius of initial circular orbit (km)
r_atm   = h_atm + RE;      %Radius of sensible atmosphere limit (km)
V0      = sqrt(MU./r0);     %Circular orbit velocity (km/s)
r_ratio = r_atm./r0;       %Radius ratio
fpa0    = deg2rad(fpa0_deg); %Flight-path angle (rad)
cfpa0   = cos(fpa0);       %Variable simplification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Descent Impulse Determination
%Equation components
Radicand_num   = 2.*(1 - r_ratio);
Radicand_denom = r_ratio.*(1 - ((r_ratio.^2).*(cfpa0.^2)));
Radicand       = Radicand_num./Radicand_denom;

%Descent trajectory impulse (km/s)
dV1 = V0.*(1 - ((r_ratio.*cfpa0).*sqrt(Radicand)));
V1 = V0.*sqrt(Radicand); %Entry velocity (km/s)
```

Eccentricity.m

```
function ecc = Eccentricity(r_apogee,r_perigee)

ecc = ((r_apogee - r_perigee)/(r_apogee + r_perigee)); %Eccentricity
```

EntryDecel.m

```
function [decel] = EntryDecel(Gravity_Choice,mass,S,Cd,Cl,r,V,lat,fpa)

global RE FlatE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planetary Model
[GravModel] = GravityModel(r,lat);
g = GravModel.g; %Spherical gravity model (km/s^2)

if Gravity_Choice == 1 %Spherical gravity model
    h_gd = r - RE;
elseif Gravity_Choice == 2 %J2 gravity model
    [h_gd,lat_gd] = Geocentric2Geodetic(r,lat,RE,FlatE);
end

%Atmospheric density (kg/km^3)
[Rho] = AtmosModel_PostAnalysis(h_gd,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Dynamics
D      = 0.5.*Rho.*Cd.*S.*(V.^2); %Drag force
L      = 0.5.*Rho.*Cl.*S.*(V.^2); %Lift force

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Deceleration
%Tangential component (along velocity vector)
decel.Tang      = (D./mass) + (g.*sin(fpa));
decel.TangG     = decel.Tang./g;    %(g's)

%Normal component (along lift vector)
decel.Normal    = -(L./mass) - (((V.^2)./r) - g).*cos(fpa);
decel.NormalG   = decel.Normal./g; %(g's)

%Magnitude of deceleration
decel.Mag       = sqrt((decel.Tang.^2) + (decel.Normal.^2));

%Magnitude of deceleration (g's)
decel.Gs        = decel.Mag./g;
```

```

function [FirstMin,FirstMax,FirstSkip_States] = FirstSkip(t,traj_states)

global RE FlatE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Variable Assignment of States (x)
%Radial position (km)
r      = traj_states(:,1);
%Velocity (km/s)
V      = traj_states(:,2);
%Longitude (rad)
lon    = traj_states(:,3);
%Latitude (rad)
lat    = traj_states(:,4);
%Flight-path angle (rad)
fpa    = traj_states(:,5);
%Heading angle (rad)
heading = traj_states(:,6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of First Perigee States in Skip Trajectory
for ctr_min = 2:length(r)
    if r(ctr_min) < r(ctr_min - 1)
        ctr_min      = ctr_min + 1;
    else
        %First perigee
        ctr_FirstMin = ctr_min - 1;          %Counter value for first perigee
        FirstMin.t   = t(ctr_FirstMin);      %Time (user-specified units)
        FirstMin.r   = r(ctr_FirstMin);      %Radial position (km)
        break
    end
end

%States associated with first perigee
FirstMin.V      = V(ctr_FirstMin);          %Velocity (km/s)
FirstMin.lon    = lon(ctr_FirstMin);        %Longitude (rad)
FirstMin.lat    = lat(ctr_FirstMin);        %Geocentric latitude (rad)
FirstMin.fpa    = fpa(ctr_FirstMin);        %Flight-path angle (rad)
FirstMin.heading = heading(ctr_FirstMin);   %Heading angle (rad)

%Geodetic altitude (km), geodetic latitude (rad) of first perigee
[FirstMin.h_gd, FirstMin.lat_gd] = ...
    Geocentric2Geodetic(FirstMin.r,FirstMin.lat,RE,FlatE);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Determination of First Apogee States in Skip Trajectory
for ctr_max = (ctr_FirstMin + 1):length(r)
    if r(ctr_max) > r(ctr_max - 1) && ctr_max < length(r)
        ctr_max      = ctr_max + 1;
    else

```



```

        %First apogee
        ctr_FirstMax = ctr_max - 1;           %Counter value for first apogee
        FirstMax.t    = t(ctr_FirstMax);      %Time (user-specified units)
        FirstMax.r    = r(ctr_FirstMax);      %Radial position (km)
        break
    end
end

%States associated with first perigee
FirstMax.V          = V(ctr_FirstMax);       %Velocity (km/s)
FirstMax.lon        = lon(ctr_FirstMax);     %Longitude (rad)
FirstMax.lat        = lat(ctr_FirstMax);     %Geocentric latitude (rad)
FirstMax.fpa        = fpa(ctr_FirstMax);     %Flight-path angle (rad)
FirstMax.heading    = heading(ctr_FirstMax); %Heading angle (rad)

%Geodetic altitude (km), geodetic latitude (rad) of first apogee
[FirstMax.h_gd, FirstMax.lat_gd] = ...
    Geocentric2Geodetic(FirstMax.r,FirstMax.lat,RE,FlatE);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% States Associated with First Skip (from Initial Descent to First Apogee)
FirstSkip_States.t    = t(1:ctr_FirstMax,1); %Time
FirstSkip_States.r    = r(1:ctr_FirstMax,1); %Radial position (km)
FirstSkip_States.V    = V(1:ctr_FirstMax,1); %Velocity (km/s)
FirstSkip_States.lon  = lon(1:ctr_FirstMax,1); %Longitude (rad)
FirstSkip_States.lat  = lat(1:ctr_FirstMax,1); %Geocentric lat. (rad)
FirstSkip_States.fpa  = fpa(1:ctr_FirstMax,1); %FPA (rad)
FirstSkip_States.heading = heading(1:ctr_FirstMax,1); %Heading angle (rad)

%Geodetic altitude (km), geodetic latitude (rad) of first skip
[FirstSkip_States.h_gd, FirstSkip_States.lat_gd] = ...
    Geocentric2Geodetic(FirstSkip_States.r,FirstSkip_States.lat,RE,FlatE);

```

Geocentric2Geodetic.m

```
function [h_gd, lat_gd] = Geocentric2Geodetic(r,lat,RE,FlatE)

rho = r/RE;

%Geodetic altitude
h_Component1 = (rho - 1);
h_Component2 = (0.5.*(1 - cos(2.*lat))).*FlatE;
h_Component3 = (((1./(4.*rho)) - (1/16)).*(1 - cos(4.*lat))).*(FlatE.^2);
h_gd = (h_Component1 + h_Component2 + h_Component3) * RE;

%Geodetic latitude
lat_Component1 = ((sin(2.*lat))./rho).*FlatE;
lat_Component2 = (((1./(rho.^2)) - (1./(4.*rho))).*sin(4.*lat)).*(FlatE.^2);
lat_gd = lat + lat_Component1 + lat_Component2;
```

Geodetic2Geocentric.m

```
function [r_gc, lat_gc] = Geodetic2Geocentric(h_gd,lat_gd,RE,FlatE)

h = h_gd/RE;

%Geocentric altitude
r_Component1 = (h + 1);
r_Component2 = (-0.5.*(1 - cos(2.*lat_gd))).*FlatE;
r_Component3 = (((1./(4.*(h+1))) + (1/16)).*(1-cos(4.*lat_gd))).*(FlatE.^2);
r_gc = (r_Component1 + r_Component2 + r_Component3) * RE;

%Geocentric latitude
lat_Component1 = ((-sin(2.*lat_gd))./(h+1)).*FlatE;
lat_Component2 = ((-sin(2.*lat_gd))./(2.*((h+1).^2)));
lat_Component3 = ((1./(4.*((h+1).^2))) + (1./(4.*(h+1)))).*sin(4.*lat_gd);
lat_gc = lat_gd + lat_Component1 + (lat_Component2 + ...
    lat_Component3).*(FlatE.^2);
```

GroundTargets.m

```
function GroundTargets

global Lat_Denver    Lon_Denver    Lat_Gibraltar Lon_Gibraltar
global Lat_Glasgow   Lon_Glasgow   Lat_Grozny   Lon_Grozny
global Lat_Moscow     Lon_Moscow    Lat_Ponti    Lon_Ponti
global Lat_Pyong      Lon_Pyong     Lat_Reyk     Lon_Reyk
global Lat_Tehran     Lon_Tehran    Lat_Tokyo    Lon_Tokyo
global Lat_Brasil     Lon_Brasil    Lat_Buenos   Lon_Buenos
global Lat_Canberra   Lon_Canberra  Lat_Cape     Lon_Cape

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Geographical Coordinates
%DENVER, United States
Lat_Denver    =    39.7392; %(deg N)
Lon_Denver    =   -104.9842; %(deg W)

%GIBRALTAR
Lat_Gibraltar =    36.1430; %(deg N)
Lon_Gibraltar =    -5.3530; %(deg W)

%GLASGOW, Scotland
Lat_Glasgow   =    55.8700; %(deg N)
Lon_Glasgow   =    -4.2700; %(deg W)

%GROZNY, Chechnya
Lat_Grozny    =    43.2983; %(deg N)
Lon_Grozny    =    45.6997; %(deg E)

%Moscow, Russia
Lat_Moscow    =    55.7517; %(deg N)
Lon_Moscow    =    37.6178; %(deg E)

%Pontianak, Indonesia
Lat_Ponti     =     0.0000; %(deg N)
Lon_Ponti     =   109.3333; %(deg E)

%Pyongyang, North Korea
Lat_Pyong     =    39.0333; %(deg N)
Lon_Pyong     =   125.7500; %(deg E)

%Reykjavik, Iceland
Lat_Reyk      =    64.1333; %(deg N)
Lon_Reyk      =   -21.9333; %(deg W)

%Tehran, Iran
Lat_Tehran    =    35.6833; %(deg N)
Lon_Tehran    =    51.4167; %(deg E)
```

```

%Tokyo, Japan
Lat_Tokyo    =    35.6833; %(deg N)
Lon_Tokyo    =    139.7667; %(deg E)

%Brasilia, Brazil
Lat_Brasil   =    -15.7810; %(deg S)
Lon_Brasil   =    -47.9196; %(deg W)

%Buenos Aires, Argentina
Lat_Buenos   =    -34.6036; %(deg S)
Lon_Buenos   =    -58.3817; %(deg W)

%Canberra, Australia
Lat_Canberra =    -35.2828; %(deg S)
Lon_Canberra =    149.1314; %(deg E)

%Cape Town, South Africa
Lat_Cape     =    -33.9767; %(deg S)
Lon_Cape     =    18.4244; %(deg E)

```

HeatFluxModel.m

```

function [HeatModel,Eta,T_KE] = HeatFluxModel(V_SI,Rho_SI,Emissivity, ...
                                              Tw_F,Tinf_F,mass,S,Cd,Cl)

global MU g0 RE BetaH Rho0 StefBoltz
WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variable/Function Simplification
Eta = ((Rho_SI.*Cd.*S)./(2*mass.*BetaH)); %Altitude (non-dimensional)
T_KE = ((V_SI.^2)./(2*g0.*RE));           %Kinetic energy (non-dimensional)
Ve   = sqrt(MU/RE);                       %Planetary surface vel. (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Average Wall Heat Flux (Non-Dimensional);           Source: Hicks (2009)
HeatModel.Qw = Eta.*((T_KE).^(3/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Average Stagnation Heat Flux (Non-Dimensional); Source: Hicks (2009)
HeatModel.Qs = ((Eta).^(1/2)).*((T_KE).^(3/2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stagnation Heat Flux (kW/m^2);                       Source: Rao (2002)
%Heating rate constant (kW/m^2)
Qdot_Bar = 17600*11.35377; %Source: Rao, et al. (2002)
% Qdot_Bar = 11.357;      %Source: Rao, et al. (2008)
% Qdot_Bar = 199870;      %Source: Darby-Rao (2010)

HeatModel.Qdot = Qdot_Bar*((Rho_SI./Rho0).^(0.5)).*((V_SI./Ve).^(3.15));

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Stagnation Heat Flux (kW/m^2);                               Source: Havey (1982)
Heat_Coeff = 17700;
Rn_SI      = 0.3048;
V          = convlength(V_SI, 'km', 'ft');                    %Velocity, (km/s)->(ft/s)
Rn         = convlength(Rn_SI, 'm', 'ft');                    %Nose radius, (m)->(ft)
Rho_m3     = Rho_SI.*(1/(1000^3));                             %Density, (kg/km^3)->(kg/m^3)
Rho        = convdensity(Rho_m3, 'kg/m^3', ...                %Density, (kg/m^3)->(sl/ft^3)
                          'slug/ft^3');
Tw_R       = convtemp(Tw_F, 'F', 'R');                        %Wall temperature, (deg F)->(R)
Tinf_R     = convtemp(Tinf_F, 'F', 'R');                      %Free-stream temp., (deg F)->(R)

hw = 0.24.*Tw_R;
h0 = (0.24.*Tinf_R) + ((V.^2)./(50063));

HeatFlux_Havey = Heat_Coeff.*((Rho./Rn).^(0.5)).* ...
                ((V./(1E4)).^(3.07)).*(1 - (hw./h0));
HeatModel.QHavey = HeatFlux_Havey.*11.35377;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Stagnation Heat Flux (kW/m^2);                               Source: Galman (1961)
%Note: Variables (V, Rn, Rho) obtained from preceding section
HeatFlux_Galman = ((2*Rn)^(-0.5)).* ...
                  ((3.18).*(Rho.^(0.5)).*(V.^(3.2)).*(1E-9));
HeatModel.QGalman = HeatFlux_Galman.*11.35377;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Radiative Heat Flux (kW/m^2)
Tinf_K = convtemp(Tinf_F, 'F', 'K'); %Free-stream temperature, (deg F)->(K)
HeatModel.Qr1 = (Emissivity.*StefBoltz.*((Tinf_K.^4)))./1000;

```

InclinationChange.m

```

function dV_Simple = InclinationChange(V0,fpa0,dIncl)

%Converts and overwrites initial angle variables
fpa0 = deg2rad(fpa0); dIncl = deg2rad(dIncl);

%Delta-V required for simple, inclination-only plane change
dV_Simple = abs(2.*V0.*cos(fpa0).*sin(0.5.*dIncl));

```

KeplerAnomalies.m

```
function [E,nu] = KeplerAnomalies(MeanAnom,ecc)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Eccentric Anomaly
%Initial guess for eccentric anomaly
if MeanAnom < 0 || MeanAnom > pi
    E0 = MeanAnom - ecc;
else
    E0 = MeanAnom + ecc;
end

IterCount = 0; %Counter initialization
IterMax    = 20; %Maximum number of iterations

%Newton-Rhapson iteration
E1 = E0 + ((MeanAnom - E0 + (ecc*sin(E0)))/(1 - (ecc*cos(E0))));

while (abs(E1 - E0) > 1E-15) && (IterCount < IterMax)
    E0 = E1;
    E1 = E0 + ((MeanAnom - E0 + (ecc*sin(E0)))/(1 - (ecc*cos(E0))));
    IterCount = IterCount + 1;
end

E = E1; %Eccentric anomaly (rad)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% True Anomaly
nu = acos((cos(E) - ecc)/(1 - (ecc*cos(E)))); %(rad)
```

OrbitVelocity.m

```
function V = OrbitVelocity(r,ecc,nu_deg)

global MU

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Orbital Velocity
nu = deg2rad(nu_deg); %True anomaly (rad)

%Equation components
V_Component1 = (MU./r);
V_Component2 = 1 - (ecc^2);
V_Component3 = 1 + (ecc*cos(nu));
V = sqrt(V_Component1.*(2 - (V_Component2./V_Component3)));
```

PayloadImager.m

```
function [FOV_m2,FOV_km2,Resolution] = PayloadImager(h,Aperture_Diameter, ...
                                                    FocalLength,lambda)

%NOTE: Altitude (h) is units of meters (m)
%Linear (ground) resolution (m)
Resolution = 2.44*(h*lambda*(1/Aperture_Diameter));

%Image plane radius (m)
ImagPlane_Radius = 0.5*Aperture_Diameter;

%Angular diameter of FOV (rad)
FOV_AngularDiam = 2*atan(ImagPlane_Radius/FocalLength);

%Ground object/FOV (m^2; km^2)
FOV_m2 = pi*((h*tan(0.5*FOV_AngularDiam))^2);
FOV_km2 = FOV_m2 * (1/(1000^2));
```

RelativeStates.m

```
function [V_Rel,PSI_Rel] = RelativeStates(mass,S,Cd,Cl,h_Orbit, ...
                                           lat,fpa,heading,bank)

global RE OmegaE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Variable/Function Simplification
%Latitude (lat), flight-path (fpa), and bank (sigma) angles
clat = cos(lat); slat = sin(lat);
cfpa = cos(fpa); sfpa = sin(fpa); cbank = cos(deg2rad(bank));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planetary Model
r = h_Orbit + RE; %Radial position (km)
[GravModel] = GravityModel(r,lat);
g = GravModel.g; %Spherical gravity model (km/s^2)

[Rho] = AtmosModel(h_Orbit,2);
rho_r = Rho;

%Planetary rotation parameter
OmegaRot = OmegaE;
OmegaRot2 = OmegaRot^2;
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Vehicle Aerodynamics
D_comp = 0.5*rho_r*Cd*S; %Drag force component
L_comp = 0.5*rho_r*Cl*S; %Lift force component

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Newton-Raphson Iteration
PSI_Guess = heading; %Initial heading angle guess (rad)
PSI_Update = 999; %Initialization of heading angle update (rad)
IterCount = 0; %Initialization of iteration counter

while abs(PSI_Update - PSI_Guess) > 1E-12
    PSI_Update = PSI_Guess; %Re-define current heading angle (rad)

    %Quadratic equation and components
    A = (1/r) + ((L_comp*cbank)/mass);
    B = 2*OmegaRot*clat*cos(PSI_Update);
    C = -(g*cfpa) + (r*OmegaRot2*clat*(clat*cfpa + ...
        slat*sin(PSI_Update)*sfpa));
    V_Check = (-B + sqrt((B^2) - (4*A*C)))/(2*A);

    %Updated heading angle (rad)
    PSI_Guess = heading + asin((2*pi*r*sin(heading))/(86400*V_Check));

    %Difference between update and guess heading angle
    IterativeDiff_PSI = abs(PSI_Update - PSI_Guess);
    IterCount = IterCount + 1; %Update to iteration counter
end

V_Rel = V_Check; %Relative maneuver velocity (km/s)
PSI_Rel = PSI_Guess; %Relative heading angle (rad)

```

RelativeStates_Entry.m

```

function [Vrel0_mag,fpa_rel0,heading_rel0] = ...
    RelativeStates_Entry(h0,V_Boost,lon0,lat0,fpa0,heading0)

global RE MU OmegaE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maneuver Profile Initial Conditions (Inertial, Unknown Epoch)
r0 = h0 + RE; %Radial position (km)
V0 = sqrt(MU/r0) + V_Boost; %Orbit velocity (km/s)

%Initial radial position and velocity vectors
r0_init = [r0; 0; 0]; %(km)
V0_init = [0; V0; 0]; %(km/s)

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Coordinate Transformations for Initial Conditions
%Radial position vector relative to inertial frame (km)
I_r0      = ROT(3,-lon0,1)*ROT(2,lat0,1)*r0_init;
I_r0_mag = norm(I_r0); %Radial position magnitude (km)

%Vehicle-pointing (VP) velocity vector relative to inertial frame (km/s)
VP_V0     = ROT(1,-heading0,1)*ROT(3,fpa0,1)*V0_init;
%Earth-fixed velocity vector relative to inertial frame (km/s)
I_V0      = ROT(3,-lon0,1) *ROT(2,lat0,1)*VP_V0;

%Planetary angular velocity vector (rad/s)
OmegaE_vec = [0; 0; OmegaE];

%Velocity vector relative to rotating planet (km/s)
Vrel0_vec = I_V0 - cross(OmegaE_vec,I_r0);
Vrel0_mag = norm(Vrel0_vec); %Relative velocity vector magnitude (km/s)

%Initial flight-path and heading angles relative to rotating planet (rad)
fpa_rel0   = (pi/2) - acos((dot(Vrel0_vec,I_r0))/(Vrel0_mag*I_r0_mag));
heading_rel0 = (pi/2) - acos((dot(Vrel0_vec,[0;0;1]))/Vrel0_mag);

```

ROT.m

```

function B = ROT(axis,angle,Angle_Choice)

%Conversion of angle from degrees to radians
if      Angle_Choice == 1 %Angle conversion NOT required
    angle = angle;
elseif Angle_Choice == 2 %Angle conversion required
    angle = deg2rad(angle);
end

%Rotation matrices
if      axis == 1 %Rotation about Axis #1
    B = [1      0      0;...
          0 cos(angle) sin(angle);...
          0 -sin(angle) cos(angle)];
elseif axis == 2 %Rotation about Axis #2
    B = [cos(angle) 0 -sin(angle);...
          0      1      0;...
          sin(angle) 0  cos(angle)];
elseif axis == 3 %Rotation about Axis #3
    B = [ cos(angle) sin(angle) 0;...
          -sin(angle) cos(angle) 0;...
          0          0      1];
end

```

SingleSkip_Maneuver.m

```
function [SingleSkip_t,SingleSkip_States] = ...
    SingleSkip_Maneuver(Choice_1,Choice_2,Choice_3,Choice_4,    ...
        Choice_5,Choice_6,Time_Max,r,V,lon,lat,    ...
        fpa,heading,bank)

%Simulates skip entry trajectory for given initial conditions
[t_vec, traj_states] = Maneuver_MainFunction(Choice_1,Choice_2, ...
    Choice_3,Choice_4,Choice_5,Choice_6,    ...
    Time_Max,r,V,lon,lat,fpa,heading,bank);

if isnan(traj_states(end,2)) == 1

%Deletes rows with 'NaN'
traj_states(isnan(traj_states(:,2)),:)=[];

%Limits time vector length to length of trajectory parameter matrix
t_vec = t_vec(1:length(traj_states(:,1)),1);

if isempty(traj_states) == 1
    SingleSkip_t = 0;
    SingleSkip_States = zeros(1,8);
    return
else
    SingleSkip_t = 0;
    SingleSkip_States = zeros(1,8);
    return
end

else
    %Determines states associated with single skip maneuver
    [FirstMin,FirstMax,FirstSkip_States] = FirstSkip(t_vec,traj_states);

    %Re-assignment of time and state vectors for single skip maneuver
    SingleSkip_t = FirstSkip_States.t;
    SingleSkip_States = [FirstSkip_States.r,    FirstSkip_States.V,    ...
        FirstSkip_States.lon, FirstSkip_States.lat,    ...
        FirstSkip_States.fpa, FirstSkip_States.heading,    ...
        FirstSkip_States.h_gd,FirstSkip_States.lat_gd];
end
```

SMARadii.m

```
function [Radii,Altitude] = SMARadii(sma,ecc)

global RE

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Radius and altitude of perigee (km)
Radii.r_perig    = sma*(1 - ecc);
Altitude.h_perig = Radii.r_perig - RE;

%Radius and altitude of apogee (km)
Radii.r_apog     = sma*(1 + ecc);
Altitude.h_apog  = Radii.r_apog - RE;
```

TimeUtility.m

```
function [t_modified,time_string] = TimeUtility(t_vector,Time_Choice)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 'Time_Choice' Options
%1 = %Retains time unit of 'seconds'
%2 = %Converts time units from 'seconds' to 'minutes'
%3 = %Converts time units from 'seconds' to 'hours'
%4 = %Converts time units from 'seconds' to 'days'

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time Conversions
if Time_Choice == 1
    t_modified = t_vector;
    time_string = '(sec)'; % 'seconds'

elseif Time_Choice == 2
    t_modified = t_vector * (1/60);
    time_string = '(min)'; % 'minutes'

elseif Time_Choice == 3
    t_modified = t_vector * (1/60) * (1/60);
    time_string = '(hr)'; % 'hours'

elseif Time_Choice == 4
    t_modified = t_vector * (1/60) * (1/60) * (1/24);
    time_string = '(day)'; % 'days'
end
```

TLE2RV.m

```
function [PositionVec,VelocityVec,h_perig,h_apog] = ...
    TLE2RV(incl_deg,raan_deg,ecc,omega_deg,MeanAnom_deg,MeanMotion_rev)

global MU

WGS84Constants; %Loads global constants from external m-file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TLE Inputs (Example)
% ecc          = 0.6887925; %Eccentricity
% incl_deg     = 63.5982;   %Inclination (deg)
% raan_deg     = 126.1576;  %Right ascension of the ascending node (deg)
% omega_deg    = 276.0005;  %Argument of perigee (deg)
% MeanAnom_deg = 122.0897;  %Mean anomaly (deg)
% MeanMotion_rev = 2.00582243; %Mean motion (rev/day)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Unit Conversion
incl      = deg2rad(incl_deg);          %(rad)
raan      = deg2rad(raan_deg);          %(rad)
omega     = deg2rad(omega_deg);         %(rad)
MeanAnom  = deg2rad(MeanAnom_deg);      %(rad)
MeanMotion = MeanMotion_rev*(2*pi)*(1/86400); %(rad/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Secondary Orbital Elements
sma      = (MU/(MeanMotion^2))^(1/3); %Semi-major axis (km)

[E,nu] = KeplerAnomalies(MeanAnom,ecc);
nu_deg = rad2deg(nu);                %True anomaly (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Position and Velocity Vectors
[r,V] = COE2RV(MU,sma,ecc,incl,raan,omega,nu,1);

PositionVec = r'; %Position vector (km)
VelocityVec = V'; %Velocity vector (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Semi-Major Axis Radii
[Radii,Altitude] = SMARadii(sma,ecc);
h_perig = Altitude.h_perig; %Perigee altitude (km)
h_apog  = Altitude.h_apog;  %Apogee altitude (km)
```

Appendix H. MATLAB® Code for Design of Experiments Support Utilities

Table H.1. m-File Classification for Design of Experiments Support Utilities

Filename	File Type	Description
DOE_BankAngle	Script	Pareto optimization with variable σ
DOE_MainEffectsPlot	Function	Plots main effects
DOE_MainEffectsPlotting	Script	Plots main effects
DOEAnalysis	Script	Primary driver of DOE simulations
ParetoBoundary_5Factors	Script	Pareto optimization with 5 factors
ParetoBoundary_6Factors	Script	Pareto optimization with 6 factors
ParetoBoundary_InitAlt	Script	Pareto optimization for h_i analysis
ParetoDOE	Script	Augmented Pareto front analysis
paretofront	Function	Determines Pareto front points

DOE_BankAngle.m

```
close all; clear all; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MIN Delta-V
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 8; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_dV = 0; %Constraint for minimum delta-V (km/s)
obj_x = 7; %Column number of x-axis objective (from reduced matrix)
obj_y = 6; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_729_80_100; %Bank angle range: [-100, -80]
load DOEMatrix_729_50_80; %Bank angle range: [ -80, -50]
load DOEMatrix_729_20_50; %Bank angle range: [ -50, -20]
load DOEMatrix_729_0_20; %Bank angle range: [ 0, -20]
load DOEMatrix_729_100_120; %Bank angle range: [-120,-100]

%% Matrix of Experiments and Observations
%Bank Angle Campaign #1
IN_1 = DOEMatrix_729_80_100(:,1:end);
split_1 = length(IN_1(:,1));
total_1 = length(IN_1(:,1)); %Length of input matrix (number of rows)
```

```

%Bank Angle Campaign #2
IN_2      = DOEMatrix_729_50_80(:,1:end);
split_2 = length(IN_2(:,1));
total_2 = length(IN_2(:,1)); %Length of input matrix (number of rows)

%Bank Angle Campaign #3
IN_3      = DOEMatrix_729_20_50(:,1:end);
split_3 = length(IN_3(:,1));
total_3 = length(IN_3(:,1)); %Length of input matrix (number of rows)

%Bank Angle Campaign #4
IN_4      = DOEMatrix_729_0_20(:,1:end);
split_4 = length(IN_4(:,1));
total_4 = length(IN_4(:,1)); %Length of input matrix (number of rows)

%Bank Angle Campaign #5
IN_5      = DOEMatrix_729_100_120(:,1:end);
split_5 = length(IN_5(:,1));
total_5 = length(IN_5(:,1)); %Length of input matrix (number of rows)

%% Creation of Reduced Factor and Observation Matrices
%Bank Angle Campaign #1
for ii = 1:size(IN_1,1)
    x_star_1(ii,:) = IN_1(ii,1:nvars); %Factors
    J_1(ii,:)      = IN_1(ii,nvars+1:size(IN_1,2)); %Observations
end

%Bank Angle Campaign #2
for jj = 1:size(IN_2,1)
    x_star_2(jj,:) = IN_2(jj,1:nvars); %Factors
    J_2(jj,:)      = IN_2(jj,nvars+1:size(IN_2,2)); %Observations
end

%Bank Angle Campaign #3
for kk = 1:size(IN_3,1)
    x_star_3(kk,:) = IN_3(kk,1:nvars); %Factors
    J_3(kk,:)      = IN_3(kk,nvars+1:size(IN_3,2)); %Observations
end

%Bank Angle Campaign #4
for kk = 1:size(IN_4,1)
    x_star_4(kk,:) = IN_4(kk,1:nvars); %Factors
    J_4(kk,:)      = IN_4(kk,nvars+1:size(IN_4,2)); %Observations
end

%Bank Angle Campaign #5
for kk = 1:size(IN_5,1)
    x_star_5(kk,:) = IN_5(kk,1:nvars); %Factors
    J_5(kk,:)      = IN_5(kk,nvars+1:size(IN_5,2)); %Observations
end

```

```

%% Determination of Observations which Satisfy Constraints
%Bank Angle Campaign #1
I_1 = find(J_1(1:split_1,obj_x) >= min_incl & ...
           J_1(1:split_1,obj_y) >= min_dV);
Z_1 = (find(J_1(split_1+1:total_1,obj_x) >= min_incl & ...
           J_1(split_1+1:total_1,obj_y) >= min_dV)+split_1);
J_filt_1 = J_1(I_1,:);
J_filt1_1 = J_1(Z_1,:);

%Bank Angle Campaign #2
I_2 = find(J_2(1:split_2,obj_x) >= min_incl & ...
           J_2(1:split_2,obj_y) >= min_dV);
Z_2 = (find(J_2(split_2+1:total_2,obj_x) >= min_incl & ...
           J_2(split_2+1:total_2,obj_y) >= min_dV)+split_2);
J_filt_2 = J_2(I_2,:);
J_filt1_2 = J_2(Z_2,:);

%Bank Angle Campaign #3
I_3 = find(J_3(1:split_3,obj_x) >= min_incl & ...
           J_3(1:split_3,obj_y) >= min_dV);
Z_3 = (find(J_3(split_3+1:total_3,obj_x) >= min_incl & ...
           J_3(split_3+1:total_3,obj_y) >= min_dV)+split_3);
J_filt_3 = J_3(I_3,:);
J_filt1_3 = J_3(Z_3,:);

%Bank Angle Campaign #4
I_4 = find(J_4(1:split_4,obj_x) >= min_incl & ...
           J_4(1:split_4,obj_y) >= min_dV);
Z_4 = (find(J_4(split_4+1:total_4,obj_x) >= min_incl & ...
           J_4(split_4+1:total_4,obj_y) >= min_dV)+split_4);
J_filt_4 = J_4(I_4,:);
J_filt1_4 = J_4(Z_4,:);

%Bank Angle Campaign #5
I_5 = find(J_5(1:split_5,obj_x) >= min_incl & ...
           J_5(1:split_5,obj_y) >= min_dV);
Z_5 = (find(J_5(split_5+1:total_5,obj_x) >= min_incl & ...
           J_5(split_5+1:total_5,obj_y) >= min_dV)+split_5);
J_filt_5 = J_5(I_5,:);
J_filt1_5 = J_5(Z_5,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factors Associated with Observations which Satisfy Constraints
%Bank Angle Campaign #1
x_star_filt_1 = x_star_1(I_1,:);
x_star_filt1_1 = x_star_1(Z_1,:);

%Bank Angle Campaign #2
x_star_filt_2 = x_star_2(I_2,:);
x_star_filt1_2 = x_star_2(Z_2,:);

%Bank Angle Campaign #3
x_star_filt_3 = x_star_3(I_3,:);
x_star_filt1_3 = x_star_3(Z_3,:);

```

```

%Bank Angle Campaign #4
x_star_filt_4 = x_star_4(I_4,:);
x_star_filt1_4 = x_star_4(Z_4,:);

%Bank Angle Campaign #5
x_star_filt_5 = x_star_5(I_5,:);
x_star_filt1_5 = x_star_5(Z_5,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting of Design Space
% subplot(1,2,1);
figure; %Bank Angle Campaign #5
scatter((J_filt_5(:,obj_x)-InitIncl)',J_filt_5(:,obj_y)./BaselineCost', ...

'SizeData',2^2,'MarkerEdgeColor','k','MarkerFaceColor','c','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');

hold on; %Bank Angle Campaign #1
scatter((J_filt_1(:,obj_x)-InitIncl)',J_filt_1(:,obj_y)./BaselineCost', ...

'SizeData',2^2,'MarkerEdgeColor','k','MarkerFaceColor','b','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');

hold on; %Bank Angle Campaign #2
scatter((J_filt_2(:,obj_x)-InitIncl)',J_filt_2(:,obj_y)./BaselineCost', ...

'SizeData',2^2,'MarkerEdgeColor','k','MarkerFaceColor','g','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');

hold on; %Bank Angle Campaign #3
scatter((J_filt_3(:,obj_x)-InitIncl)',J_filt_3(:,obj_y)./BaselineCost', ...

'SizeData',2^2,'MarkerEdgeColor','k','MarkerFaceColor','r','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');

hold on; %Bank Angle Campaign #4
scatter((J_filt_4(:,obj_x)-InitIncl)',J_filt_4(:,obj_y)./BaselineCost', ...

'SizeData',2^2,'MarkerEdgeColor','k','MarkerFaceColor','m','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');
hold on;

legend('\it\sigma\rm = [-120,-100] deg', ...
       '\it\sigma\rm = [-100, -80] deg', ...
       '\it\sigma\rm = [ -80, -50] deg', ...
       '\it\sigma\rm = [ -50, -20] deg', ...
       '\it\sigma\rm = [ -20,  0] deg','location','NorthEast');

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot      = [J_filt_1; J_filt1_1; ...
                  J_filt_2; J_filt1_2; ...
                  J_filt_3; J_filt1_3; ...
                  J_filt_4; J_filt1_4; ...
                  J_filt_5; J_filt1_5];

x_star_filt_tot = [x_star_filt_1; x_star_filt1_1; ...
                  x_star_filt_2; x_star_filt1_2; ...
                  x_star_filt_3; x_star_filt1_3; ...
                  x_star_filt_4; x_star_filt1_4; ...
                  x_star_filt_5; x_star_filt1_5];

K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)]))==1);

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
        J_filt_tot(K,obj_y)./BaselineCost', 'y', ...
        'SizeData',10^2, 'Marker', 'o', ...
        'MarkerEdgeColor', 'k', 'LineWidth',1.5, 'HandleVisibility', 'off');

x_star_pareto = x_star_filt_tot(K,:);
J_pareto      = J_filt_tot(K,:);
[B IX]        = sort(J_pareto,1);
J_sorted1     = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted1     = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto_xJ     = [x_sorted1,J_sorted1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf, 'Color', 'w'); %Sets overall figure background color to 'white'
set(gcf, 'Position', get(0, 'Screensize')); %Automatically maximizes plot window

```

DOE_MainEffectsPlot.m

```
function [figh,axesh] = DOE_MainEffectsPlot(y,group,varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Use: [figh,axesh] = DOE_MainEffectsPlot(y,group,varargin)
%
% Displays main effects plots for the group means of matrix Y with groups
% defined by entries in the cell array GROUP. Y is a numeric matrix or
% vector. If Y is a matrix, the rows represent different observations and
% the columns represent replications of each observation.
%
% Author/Date      : The MathWorks, Inc./2006-2010
% Modified by     : Bettinger, Robert AFIT/ENY/2013
%
% Example:
%   Display main effects plots for car weight with two grouping variables,
%   model year and number of cylinders:
%       load carsmall;
%       maineffectsplot(Weight,{Model_Year,Cylinders}, ...
%                       'varnames',{'Model Year', '# of Cylinders'})
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin <2
    error(message('stats:maineffectsplot:FewInput'))
end

%Parse parameter/value pairs
args      = {'varnames','statistics','parent'};
defaults = {'','mean',[]};
[eid emsg varnames,statistics,parent] = ...
    internal.stats.getargs(args,defaults,varargin{:});
if ~isempty(eid)
    error(sprintf('stats:maineffectsplot:%s',eid),emsg);
end
if ~iscell(varnames) && ~ischar(varnames)
    error(message('stats:maineffectsplot:BadVarnames'));
end
if ~(ischar(varnames) || iscellstr(varnames))
    error(message('stats:maineffectsplot:BadVarnames'));
end
needvarnames = isempty(varnames);

%Character matrix grouping variable names are converted into cell array
if ischar(varnames) && ~needvarnames
    varnames = cellstr(varnames);
end

if ~ischar(statistics) || (~strcmp(statistics,'mean') ...
    && ~strcmp(statistics,'std'))
    error(message('stats:maineffectsplot:BadStatistics'));
end
```

```

plotstddev = strcmp(statistics,'std');
if plotstddev && size(y,2)==1
    error(message('stats:maineffectsplot:BadYstatistics'))
end

% Convert the GROUP to cell arrays
if isnumeric(group) %Numerical arrays
    group = num2cell(group,1);
elseif ischar(group) %Character matrix
    group = {cellstr(group)};
elseif ~iscell(group)
    group = {group}; %Possible categorical variable
end

group = group(:);
ng = length(group); %Number of grouping factors

%Convert numeric cells or character matrix to string cell array
for i = 1:ng
    if ischar(group{i})
        group{i} = cellstr(group{i});
    end
end

%Grouping variable should have the same number of items as Y
if any(cellfun(@length,group)~=size(y,1))
    error(message('stats:maineffectsplot:BadGroup'))
end

%Generate default varnames
if needvarnames
    varnames = strcat({'X'},num2str((1:ng)','%d'));
end

%The length of varnames should be the same as the number of groups
if ng ~= length(varnames)
    error(message('stats:maineffectsplot:MismatchVarnameGroup'))
end

if plotstddev
    y = nanstd(y,0,2);
end

if size(y,2) ~= 1
    y = nanmean(y,2);
end

if feature('HGUsingMATLABClasses')
    H = hg2.SceneNode.empty;
else

```

```

    H = zeros(ng,1);
end

if isempty(parent)
    parent = clf;
end

ylim = zeros(ng,2);

for i = 1:ng
    [maineffect, gname] = grpstats(y,group{i},{ 'mean', 'gname' });
    maineffect = nanmean(maineffect,2);
    H(i) = subplot(1,ng,i, 'parent',parent);
    plot(H(i),1:length(maineffect),maineffect,'.')
    set(H(i), 'xtick',1:length(maineffect))
    set(H(i), 'xticklabel',gname)
    xlabel(H(i),varnames{i})
    axis(H(i), 'tight');
    xlim(H(i),[0.5, length(maineffect)+.5]);
    ylim(i,:) = get(H(i), 'ylim');
    xlim([0.5, length(maineffect)+.5]);
end

%Re-scale y-axis limit and leave gaps between data and axes
ylimmin = min(ylim(:,1)); ylimmax = max(ylim(:,2));
df = .05*(ylimmax-ylimmin);
set(H, 'YLim',[ylimmin-df ylimmax+df]);
set(H(2:end), 'yticklabel', '');

if plotstddev
    ylabel(H(1), 'standard deviation')
else
    ylabel(H(1), 'mean')
end

if nargout>0
    figh = parent;
end

if nargout>1
    axesh = H;
end

```

DOE_MainEffectsPlotting.m

```
clear all; clc; close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Loads Experiments and Observations
%TAV mass (kg)
load MainEffect_3125mass;
%Planform area (m^2)
load MainEffect_3125PA;
%Drag coefficient
load MainEffect_3125Cd;
%Lift coefficient
load MainEffect_3125Cl;
%Perigee altitude (km)
load MainEffect_3125Perig;
%Bank angle (deg)
load MainEffect_729Bank;

InitIncl    = 37.843;          %Initial inclination (deg)
Mass_Range  = [0:1:8000]';    %TAV mass range (kg)
PA_Range    = [0:0.1:30]';    %Planform area range (m^2)
Cd_Range    = [0:0.05:3.0]';  %Drag coefficient range
Cl_Range    = [0:0.05:3.0]';  %Lift coefficient range
Perig_Range = [60:0.1:120]';  %Perigee altitude range (km)
Bank_Range  = [-120:0.1:0]';  %Bank angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TAV Mass (kg) v. Maximum Inclination Change (deg)
subplot(2,3,1);
%Main effect for experiment campaign
plot(MainEffect_3125mass_x,(MainEffect_3125mass_y - InitIncl),'bo');

%Polynomial fit for experiment campaign
[fit_3125mass,S_3125mass] = polyfit(MainEffect_3125mass_x, ...
                                   (MainEffect_3125mass_y - InitIncl),2);
[f_3125mass, delta_3125mass] = polyconf(fit_3125mass, ...
                                         Mass_Range,S_3125mass, ...
                                         'simopt','on','predopt','curve');

hold on;
h1 = plot(Mass_Range,f_3125mass,'-b');

% %Plotting of 95% confidence bounds
% hold on; plot(Mass_Range,f_3125mass + delta_3125mass,':b');
% hold on; plot(Mass_Range,f_3125mass - delta_3125mass,':b');

xlim([2000 6000]);
ylim([0 1]);
xlabel('TAV Mass, kg');
ylabel('Maximum Inclination Change (Mean Response)');
legend(h1,{'Polynomial Fit, Degree: 2'},'location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination Change (deg)
subplot(2,3,2);
%Main effect for experiment campaign
plot(MainEffect_3125PA_x,(MainEffect_3125PA_y - InitIncl),'bo');

%Polynomial fit for experiment campaign
[fit_3125PA,S_3125PA] = polyfit(MainEffect_3125PA_x, ...
                               (MainEffect_3125PA_y - InitIncl),1);
[f_3125PA, delta_3125PA] = polyconf(fit_3125PA, ...
                                     PA_Range,S_3125PA, ...
                                     'simopt','on','predopt','curve');

hold on;
h2 = plot(PA_Range,f_3125PA,'-b');

% %Plotting of 95% confidence bounds
% hold on; plot(PA_Range,f_3125PA + delta_3125PA,':b');
% hold on; plot(PA_Range,f_3125PA - delta_3125PA,':b');

xlim([15 22]);
ylim([0 1]);
xlabel('Planform Area, m^2');
ylabel('Maximum Inclination Change (Mean Response)');
legend(h2,{'Polynomial Fit, Degree: 1'},'location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Drag Coefficient v. Maximum Inclination Change (deg)
subplot(2,3,3);
%Main effect for experiment campaign
plot(MainEffect_3125Cd_x,(MainEffect_3125Cd_y - InitIncl),'bo');

%Polynomial fit for experiment campaign
[fit_3125Cd,S_3125Cd] = polyfit(MainEffect_3125Cd_x, ...
                                (MainEffect_3125Cd_y - InitIncl),3);
[f_3125Cd, delta_3125Cd] = polyconf(fit_3125Cd, ...
                                     Cd_Range,S_3125Cd, ...
                                     'simopt','on','predopt','curve');

hold on;
h3 = plot(Cd_Range,f_3125Cd,'-b');

% %Plotting of 95% confidence bounds
% hold on; plot(Cd_Range,f_3125Cd + delta_3125Cd,':b');
% hold on; plot(Cd_Range,f_3125Cd - delta_3125Cd,':b');

xlim([0.5 2.2]);
ylim([0 1]);
xlabel('Drag Coefficient');
ylabel('Maximum Inclination Change (Mean Response)');
legend(h3,{'Polynomial Fit, Degree: 3'},'location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Lift Coefficient v. Maximum Inclination Change (deg)
subplot(2,3,4);
%Main effect for experiment campaign
plot(MainEffect_3125Cl_x,(MainEffect_3125Cl_y - InitIncl),'bo');

%Polynomial fit for experiment campaign
[fit_3125Cl,S_3125Cl] = polyfit(MainEffect_3125Cl_x, ...
                               (MainEffect_3125Cl_y - InitIncl),2);
[f_3125Cl, delta_3125Cl] = polyconf(fit_3125Cl, ...
                                    Cl_Range,S_3125Cl, ...
                                    'simopt','on','predopt','curve');

hold on;
h4 = plot(Cl_Range,f_3125Cl,'-b');

% %Plotting of 95% confidence bounds
% hold on; plot(Cl_Range,f_3125Cl + delta_3125Cl,':b');
% hold on; plot(Cl_Range,f_3125Cl - delta_3125Cl,':b');

xlim([0.5 3.0]);
ylim([0 1]);
xlabel('Lift Coefficient');
ylabel('Maximum Inclination Change (Mean Response)');
legend(h4,{'Polynomial Fit, Degree: 2'},'location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination Change (deg)
subplot(2,3,5);
%Main effect for experiment campaign
plot(MainEffect_3125Perig_x,(MainEffect_3125Perig_y - InitIncl),'bo');

%Polynomial fit for experiment campaign
[fit_3125Perig,S_3125Perig] = polyfit(MainEffect_3125Perig_x, ...
                                       (MainEffect_3125Perig_y -
                                       InitIncl),3);
[f_3125Perig, delta_3125Perig] = polyconf(fit_3125Perig, ...
                                           Perig_Range,S_3125Perig, ...
                                           'simopt','on','predopt','curve');

hold on;
h5 = plot(Perig_Range,f_3125Perig,'-b');

% %Plotting of 95% confidence bounds
% hold on; plot(Perig_Range,f_3125Perig + delta_3125Perig,':b');
% hold on; plot(Perig_Range,f_3125Perig - delta_3125Perig,':b');

xlim([79 110]);
xlabel('Perigee Altitude, km');
ylabel('Maximum Inclination Change (Mean Response)');
legend(h5,{'Polynomial Fit, Degree: 3'},'location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Bank Angle (deg) v. Maximum Inclination Change (deg)
% subplot(2,3,6);
% %Main effect for experiment campaign
% plot(MainEffect_3125Bank_x,(MainEffect_3125Bank_y - InitIncl),'bo');
%
% %Polynomial fit for experiment campaign
% [fit_3125Bank,S_3125Bank] = polyfit(MainEffect_3125Bank_x, ...
%                                     (MainEffect_3125Bank_y - InitIncl),4);
% [f_3125Bank, delta_3125Bank] = polyconf(fit_3125Bank, ...
%                                         Bank_Range,S_3125Bank, ...
%                                         'simopt','on','predopt','curve');
%
% hold on;
% h6 = plot(Bank_Range,f_3125Bank,'-b');
%
% % %Plotting of 95% confidence bounds
% % hold on; plot(Bank_Range,f_3125Bank + delta_3125Bank,':b');
% % hold on; plot(Bank_Range,f_3125Bank - delta_3125Bank,':b');
%
% xlim([-120 0]);
% % ylim([-1 10]);
% xlabel('Bank Angle, deg');
% ylabel('Maximum Inclination Change (Mean Response)');
% legend(h6,{ 'Polynomial Fit, Degree: 4' },'location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

DOEAnalysis.m

```

clear all; clc; close all;

Factor_Choice = 2;
%1 = 5-factor experiments with constant Bank Angle
%2 = 6-factor experiments with variable Bank Angle

ReCirc_Choice = 1;
%1 = Re-circularization at Skip Apogee
%2 = Re-circularization via Hohmann Transfer at 500 km if Apogee < 500 km

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Loads Orthogonal Matrix of Experiments
DOEMatrix = ...
[2000    18.5     0.5  3.0  86.75    1000    -90];

```



```

%Factor design values
if Factor_Choice == 1 %Constant bank angle
    Factor_mass = DOEMatrix(:,1); %Mass (kg)
    Factor_S = DOEMatrix(:,2); %Planform area (m^2)
    Factor_Cd = DOEMatrix(:,3); %Drag coefficient
    Factor_Cl = DOEMatrix(:,4); %Lift coefficient
    Factor_Perig = DOEMatrix(:,5); %Perigee altitude (km)
    Factor_InitAlt = DOEMatrix(:,6); %Initial altitude (km)
    bank_Skip = -90; %Skip maneuver bank angle (deg)

elseif Factor_Choice == 2 %Variable bank angle
    Factor_mass = DOEMatrix(:,1); %Mass (kg)
    Factor_S = DOEMatrix(:,2); %Planform area (m^2)
    Factor_Cd = DOEMatrix(:,3); %Drag coefficient
    Factor_Cl = DOEMatrix(:,4); %Lift coefficient
    Factor_Perig = DOEMatrix(:,5); %Perigee altitude (km)
    Factor_InitAlt = DOEMatrix(:,6); %Initial altitude (km)
    Factor_Bank = DOEMatrix(:,7); %Bank angle (deg)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Orbit Conditions
Vehicle_Choice = 99; %TAV selection
Target_Choice = 2; %Target selection
lon_Ref = 0; %Initial longitude (deg)
PSI_Ref = 37.843; %Heading angle (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Execution of Experiments
mm = 0; %Initializes loop index at zero
nn = 1; %Initializes vector concatenation counter at one

if ReCirc_Choice == 1 %Re-circularization at Skip Apogee

for mm = 1:length(DOEMatrix(:,1))
    %Analysis function for experiments
    if Factor_Choice == 1 %Constant bank angle
        [Trajectory_Analysis(nn,1:7), MaxIncl(nn,1), ...
        Deceleration_Analysis(nn,1:6),HeatFlux_Analysis(nn,1:5)] = ...
        BankManeuvers_fxnDOE(Vehicle_Choice,Target_Choice, ...
        lon_Ref,PSI_Ref,bank_Skip, ...
        Factor_mass(mm,1), Factor_S(mm,1), ...
        Factor_Cd(mm,1), Factor_Cl(mm,1), ...
        Factor_Perig(mm,1),Factor_InitAlt(mm,1));

    elseif Factor_Choice == 2 %Variable bank angle
        [Trajectory_Analysis(nn,1:7), MaxIncl(nn,1), ...
        Deceleration_Analysis(nn,1:6),HeatFlux_Analysis(nn,1:5)] = ...
        BankManeuvers_fxnDOE(Vehicle_Choice,Target_Choice, ...
        lon_Ref,PSI_Ref,Factor_Bank(mm,1), ...
        Factor_mass(mm,1), Factor_S(mm,1), ...
        Factor_Cd(mm,1), Factor_Cl(mm,1), ...
        Factor_Perig(mm,1),Factor_InitAlt(mm,1));
    end
end

```

```

    fprintf('Experiment #%d Completed\n',mm);
    mm = mm + 1; %Update to index counter
    nn = nn + 1; %Update to solution matrix concatenation counter
end

elseif ReCirc_Choice == 2 %Re-circularization with Hohmann Transfer
    for mm = 1:length(DOEMatrix(:,1))
        %Analysis function for experiments
        if Factor_Choice == 1 %Constant bank angle
            [Trajectory_Analysis(nn,1:7), MaxIncl(nn,1), ...
            Deceleration_Analysis(nn,1:6),HeatFlux_Analysis(nn,1:5)] = ...
            BankManeuvers_fxnDOE_Hohmann(Vehicle_Choice,Target_Choice, ...
            lon_Ref,PSI_Ref,bank_Skip, ...
            Factor_mass(mm,1), Factor_S(mm,1), ...
            Factor_Cd(mm,1), Factor_Cl(mm,1), ...
            Factor_Perig(mm,1),Factor_InitAlt(mm,1));

            elseif Factor_Choice == 2 %Variable bank angle
                [Trajectory_Analysis(nn,1:7), MaxIncl(nn,1), ...
                Deceleration_Analysis(nn,1:6),HeatFlux_Analysis(nn,1:5)] = ...
                BankManeuvers_fxnDOE_Hohmann(Vehicle_Choice,Target_Choice, ...
                lon_Ref,PSI_Ref,Factor_Bank(mm,1), ...
                Factor_mass(mm,1), Factor_S(mm,1), ...
                Factor_Cd(mm,1), Factor_Cl(mm,1), ...
                Factor_Perig(mm,1),Factor_InitAlt(mm,1));

            end

        fprintf('Experiment #%d Completed\n',mm);
        mm = mm + 1; %Update to index counter
        nn = nn + 1; %Update to solution matrix concatenation counter
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DOE Trajectory Solutions
%Columns # 1- 6: m|S|Cd|Cl|h_Perig (Commanded)|h_Init (DOE Factors)
%Columns # 7-13: Bank Angle|h_Perig (Simulated)|h_Prop|h_Prop(end)| ...
%               TimeMaxIncl|dV_Maneuver|dV_SkipTotal
%Columns #14-14: Maximum Inclination
%Columns #15-20: TangDecelG_Max|TangDecelG_Min|NormDecelG_Max| ...
%               NormDecelG_Min|MagDecelG_Max|MagDecelG_Min
%Columns #21-25: Qw_Max|Qs_Max|Qdot_Max|QHavey_Max|QGalman_Max

DOEResults = [DOEMatrix,Trajectory_Analysis,MaxIncl, ...
              Deceleration_Analysis,HeatFlux_Analysis];

```

```
close all; clear all; clc;

Pareto_Choice = 3;
%1 = MAX Delta-Inclination, MIN Delta-V
%2 = MAX Delta-Inclination, MAX Re-Circularization Altitude
%3 = MIN Delta-V, MAX Re-Circularization Altitude

Pareto_Intersect_Choice = 1;
%1 = Identifies, plots, and saves common Pareto optimal points
%2 = Converse of Choice #1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Pareto_Choice == 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MIN Delta-V
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 6; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_dV = 0; %Constraint for minimum delta-V (km/s)
obj_x = 9; %Column number of x-axis objective (from reduced matrix)
obj_y = 8; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_5_79;
%Matrix of experiments and observations
IN = DOEMatrix_3125_5_79(:,1:end);
split = length(IN(:,1));
total = length(IN(:,1)); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_dV);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_dV)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);
```

```

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
        J_filt_tot(K,obj_y)./BaselineCost','y', ...
        'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
        'MarkerEdgeColor','k','LineWidth',1,'HandleVisibility','off');
R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted1 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted1 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto_xJ = [x_sorted1,J_sorted1];

%Saves Pareto points to .MAT file
savefile = 'ParetoPoints_3125.mat';
save(savefile,'x_sorted1','J_sorted1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted1(:,2); %Mass (kg)
S_Pareto = x_sorted1(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted1(:,4); %Drag coefficient
Cl_Pareto = x_sorted1(:,5); %Lift coefficient
Perig_Pareto = x_sorted1(:,6); %Perigee altitude (km)
BC_Pareto = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted1(:,4); %Re-circularization altitude (km)
dV_Total_Pareto = J_sorted1(:,8); %Total delta-V (km/s)
MaxIncl_Pareto = J_sorted1(:,9); %Maximum inclination (deg)

```

```

dIncl_Pareto      = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range       = [0:0.1:16]';           %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'o','MarkerFaceColor','r', ...
      'MarkerEdgeColor','k','LineWidth',1);

% %Polynomial fit
% [fit_mass,S_mass]      = polyfit(dIncl_Pareto,mass_Pareto,4);
% [f_mass,  delta_mass] = polyconf(fit_mass,dIncl_Range,S_mass, ...
%                                'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_mass,'-b');
% hold on; plot(dIncl_Range,f_mass + delta_mass,':b');
% hold on; plot(dIncl_Range,f_mass - delta_mass,':b');

ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','NorthEast');
legend('Pareto Optimal Points','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r. ');

% %Polynomial fit
% [fit_PA,S_PA]          = polyfit(dIncl_Pareto,S_Pareto,4);
% [f_PA,  delta_PA]     = polyconf(fit_PA,dIncl_Range,S_PA, ...
%                                'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_PA,'-b');
% hold on; plot(dIncl_Range,f_PA + delta_PA,':b');
% hold on; plot(dIncl_Range,f_PA - delta_PA,':b');

ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','SouthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Drag Coefficient v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
plot(dIncl_Pareto,Cd_Pareto,'r. ');

% %Polynomial fit
% [fit_Cd,S_Cd]          = polyfit(dIncl_Pareto,Cd_Pareto,4);
% [f_Cd,  delta_Cd]     = polyconf(fit_Cd,dIncl_Range,S_Cd, ...
%                                'simopt','on','predopt','curve');

```

```

% hold on; plot(dIncl_Range,f_Cd,'-b');
% hold on; plot(dIncl_Range,f_Cd + delta_Cd,':b');
% hold on; plot(dIncl_Range,f_Cd - delta_Cd,':b');

ylim([0.0 3.0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Drag Coefficient');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','NorthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Lift Coefficient v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Cl_Pareto,'r. ');

% %Polynomial fit
% [fit_Cl,S_Cl] = polyfit(dIncl_Pareto,Cl_Pareto,4);
% [f_Cl, delta_Cl] = polyconf(fit_Cl,dIncl_Range,S_Cl, ...
%                             'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_Cl,'-b');
% hold on; plot(dIncl_Range,f_Cl + delta_Cl,':b');
% hold on; plot(dIncl_Range,f_Cl - delta_Cl,':b');

ylim([0.0 4.0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Lift Coefficient');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','SouthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r. ');

% %Polynomial fit
% [fit_Perig,S_Perig] = polyfit(dIncl_Pareto,Perig_Pareto,4);
% [f_Perig, delta_Perig] = polyconf(fit_Perig,dIncl_Range,S_Perig, ...
%                                  'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_Perig,'-b');
% hold on; plot(dIncl_Range,f_Perig + delta_Perig,':b');
% hold on; plot(dIncl_Range,f_Perig - delta_Perig,':b');

ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','SouthWest');
legend('Boundary Data','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
```

%
 % %Polynomial fit
 % [fit_BC,S_BC] = polyfit(dIncl_Pareto,BC_Pareto,4);
 % [f_BC, delta_BC] = polyconf(fit_BC,dIncl_Range,S_BC, ...
 % 'simopt','on','predopt','curve');
 % hold on; plot(dIncl_Range,f_BC,'-b');
 % hold on; plot(dIncl_Range,f_BC + delta_BC,':b');
 % hold on; plot(dIncl_Range,f_BC - delta_BC,':b');
 %
 % % ylim([-0.1 0.1]);
 % xlabel('Maximum Inclination Change, deg');
 % ylabel('Ballistic Coefficient, m^2/kg');
 % legend('Boundary Data','Polynomial Fit, Degree: 4', ...
 % '95% Confidence Bounds','location','NorthEast');

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif Pareto_Choice == 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MAX Re-Circ. Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 6; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_alt = 130; %Constraint for re-circularization altitude (km)
obj_x = 9; %Column number of x-axis objective (from reduced matrix)
obj_y = 4; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_5_79;

%Matrix of experiments and observations
IN = DOEMatrix_3125_5_79(:,1:end);
split = length(IN);
total = length(IN); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end

```

```

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
        J_filt_tot(K,obj_y)./BaselineCost','y', ...
        'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
        'MarkerEdgeColor','k','LineWidth',1,'HandleVisibility','off');

if Pareto_Intersect_Choice == 1
    %Loads and plots Pareto points from {max(delta-i),min(delta-V)} analysis
    load ParetoPoints_3125.mat;
    Pareto1 = [x_sorted1,J_sorted1];
    hold on; plot((J_sorted1(:,9) - InitIncl), ...
        J_sorted1(:,4),'gs','LineWidth',2);
end

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted2 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted2 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto2 = [x_sorted2,J_sorted2];

```



```

if Pareto_Intersect_Choice == 1
    %Common Pareto points between
    % {max(delta-i),min(delta-V)} and {max(delta-i),max(h_recirc)}
    Pareto_Intersect_12 = intersect(Pareto1,Pareto2,'rows');

    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_12.mat';
    save(savefile,'Pareto_Intersect_12');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted2(:,2); %Mass (kg)
S_Pareto = x_sorted2(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted2(:,4); %Drag coefficient
Cl_Pareto = x_sorted2(:,5); %Lift coefficient
Perig_Pareto = x_sorted2(:,6); %Perigee altitude (km)
BC_Pareto = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted2(:,4); %Re-circularization altitude (km)
dV_Total_Pareto = J_sorted2(:,8); %Total delta-V (km/s)
MaxIncl_Pareto = J_sorted2(:,9); %Maximum inclination (deg)
dIncl_Pareto = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range = [0:0.1:16]'; %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'o','MarkerFaceColor','r', ...
     'MarkerEdgeColor','k','LineWidth',1);

% %Polynomial fit
% [fit_mass,S_mass] = polyfit(dIncl_Pareto,mass_Pareto,4);
% [f_mass, delta_mass] = polyconf(fit_mass,dIncl_Range,S_mass, ...
%                               'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_mass,'-b');
% hold on; plot(dIncl_Range,f_mass + delta_mass,':b');
% hold on; plot(dIncl_Range,f_mass - delta_mass,':b');

ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','NorthWest');
legend('Pareto Optimal Points','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r.');
```



```

% %Polynomial fit
% [fit_PA,S_PA] = polyfit(dIncl_Pareto,S_Pareto,4);
% [f_PA, delta_PA] = polyconf(fit_PA,dIncl_Range,S_PA, ...
%                               'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_PA,'-b');
% hold on; plot(dIncl_Range,f_PA + delta_PA,':b');
% hold on; plot(dIncl_Range,f_PA - delta_PA,':b');
```



```

ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%         '95% Confidence Bounds','location','NorthWest');
legend('Boundary Data','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Drag Coefficient v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
plot(dIncl_Pareto,Cd_Pareto,'r.');
```



```

% %Polynomial fit
% [fit_Cd,S_Cd] = polyfit(dIncl_Pareto,Cd_Pareto,1);
% [f_Cd, delta_Cd] = polyconf(fit_Cd,dIncl_Range,S_Cd, ...
%                               'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_Cd,'-b');
% hold on; plot(dIncl_Range,f_Cd + delta_Cd,':b');
% hold on; plot(dIncl_Range,f_Cd - delta_Cd,':b');
```



```

ylim([0.0 3.0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Drag Coefficient');
% legend('Pareto Data','Polynomial Fit, Degree: 1', ...
%         '95% Confidence Bounds','location','NorthEast');
legend('Boundary Data','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Lift Coefficient v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Cl_Pareto,'r.');
```



```

% %Polynomial fit
% [fit_Cl,S_Cl] = polyfit(dIncl_Pareto,Cl_Pareto,4);
% [f_Cl, delta_Cl] = polyconf(fit_Cl,dIncl_Range,S_Cl, ...
%                               'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_Cl,'-b');
% hold on; plot(dIncl_Range,f_Cl + delta_Cl,':b');
% hold on; plot(dIncl_Range,f_Cl - delta_Cl,':b');
```

```

ylim([0.0 4.0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Lift Coefficient');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','SouthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r.');
```



```

% %Polynomial fit
% [fit_Perig,S_Perig] = polyfit(dIncl_Pareto,Perig_Pareto,4);
% [f_Perig, delta_Perig] = polyconf(fit_Perig,dIncl_Range,S_Perig, ...
%                                'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_Perig,'-b');
% hold on; plot(dIncl_Range,f_Perig + delta_Perig,':b');
% hold on; plot(dIncl_Range,f_Perig - delta_Perig,':b');
```



```

ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','SouthWest');
legend('Boundary Data','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
```



```

%
% %Polynomial fit
% [fit_BC,S_BC] = polyfit(dIncl_Pareto,BC_Pareto,4);
% [f_BC, delta_BC] = polyconf(fit_BC,dIncl_Range,S_BC, ...
%                             'simopt','on','predopt','curve');
% hold on; plot(dIncl_Range,f_BC,'-b');
% hold on; plot(dIncl_Range,f_BC + delta_BC,':b');
% hold on; plot(dIncl_Range,f_BC - delta_BC,':b');
```



```

%
% % ylim([-0.1 0.1]);
% xlabel('Maximum Inclination Change, deg');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

elseif Pareto_Choice == 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pareto Optimization: MIN Delta-V, MAX Re-Circularization Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars     = 6;      %Number of columns for experiment number and factors
min_dV    = 0;      %Constraint for minimum inclination (deg)
min_alt   = 130;    %Constraint for re-circularization altitude (km)
obj_x     = 8;      %Column number of x-axis objective (from reduced matrix)
obj_y     = 4;      %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1;   %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_5_79;
%Matrix of experiments and observations
IN      = DOEMatrix_3125_5_79(:,1:end);
split   = length(IN);
total   = length(IN); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:)      = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_dV & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_dV & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x))',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Total \it\Delta V\rm, km/s');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

```

```

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x))',
        J_filt_tot(K,obj_y)./BaselineCost','y',
        'SizeData',5^2,'MarkerFaceColor','r','Marker','o',
        'MarkerEdgeColor','k','LineWidth',0.5,'HandleVisibility','off');

if Pareto_Intersect_Choice == 1
    %Loads/plots common Pareto points between {max(delta-i),min(delta-V)}
    %and {max(delta-i),max(h_recirc)}
    load Pareto_Intersect_12.mat;
    hold on; plot(Pareto_Intersect_12(:,14), ...
        Pareto_Intersect_12(:,10),'gs','LineWidth',2);
end

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted3 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted3 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto3 = [x_sorted3,J_sorted3];

if Pareto_Intersect_Choice == 1

    %Common Pareto points between {max(delta-i),min(delta-V)},
    %{max(delta-i),max(h_recirc)}, and {min(delta-V),max(h_recirc)}
    Pareto_Intersect_123 = intersect(Pareto_Intersect_12,Pareto3,'rows');

    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_123.mat';
    save(savefile,'Pareto_Intersect_123');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted3(:,2); %Mass (kg)
S_Pareto = x_sorted3(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted3(:,4); %Drag coefficient
Cl_Pareto = x_sorted3(:,5); %Lift coefficient
Perig_Pareto = x_sorted3(:,6); %Perigee altitude (km)
BC_Pareto = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted3(:,4); %Re-circularization altitude (km)
dV_Total_Pareto = J_sorted3(:,8); %Total delta-V (km/s)
MaxIncl_Pareto = J_sorted3(:,9); %Maximum inclination (deg)

```

```

dIncl_Pareto      = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range       = [0:0.1:16]';           %Delta-inclination angle range (deg)
dV_Range          = [0:0.0001:0.5]; %Delta-V range (km/s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Total Delta-V (km/s)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'o','MarkerFaceColor','r', ...
      'MarkerEdgeColor','k','LineWidth',1);

% %Polynomial fit
% [fit_mass,S_mass] = polyfit(dV_Total_Pareto,mass_Pareto,4);
% [f_mass, delta_mass] = polyconf(fit_mass,dV_Range,S_mass, ...
%                                'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_mass,'-b');
% hold on; plot(dV_Range,f_mass + delta_mass,':b');
% hold on; plot(dV_Range,f_mass - delta_mass,':b');

ylim([1000 8000]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('TAV Mass, kg');
% legend('Pareto Data','Polynomial Fit, Degree: 4',...
%        '95% Confidence Bounds','location','NorthWest');
legend('Pareto Optimal Points','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Total Delta-V (km/s)
subplot(2,3,3); box on; grid off;
plot(dV_Total_Pareto,S_Pareto,'r.');
```

```

% %Polynomial fit
% [fit_PA,S_PA] = polyfit(dV_Total_Pareto,S_Pareto,4);
% [f_PA, delta_PA] = polyconf(fit_PA,dV_Range,S_PA, ...
%                             'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_PA,'-b');
% hold on; plot(dV_Range,f_PA + delta_PA,':b');
% hold on; plot(dV_Range,f_PA - delta_PA,':b');

ylim([10 25]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Planform Area, m^2');
% legend('Pareto Data','Polynomial Fit, Degree: 4',...
%        '95% Confidence Bounds','location','SouthWest');
legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Drag Coefficient v. Total Delta-V (km/s)
subplot(2,3,4); box on; grid off;
plot(dV_Total_Pareto,Cd_Pareto,'r.');
```

```

% %Polynomial fit
% [fit_Cd,S_Cd] = polyfit(dV_Total_Pareto,Cd_Pareto,4);
% [f_Cd, delta_Cd] = polyconf(fit_Cd,dV_Range,S_Cd, ...
%                             'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_Cd,'-b');
```

```

% hold on; plot(dV_Range,f_Cd + delta_Cd,':b');
% hold on; plot(dV_Range,f_Cd - delta_Cd,':b');

ylim([0.0 3.0]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Drag Coefficient');
% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%         '95% Confidence Bounds','location','NorthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Lift Coefficient v. Total Delta-V (km/s)
subplot(2,3,5); box on; grid off;
plot(dV_Total_Pareto,Cl_Pareto,'r. ');

% %Polynomial fit
% [fit_Cl,S_Cl] = polyfit(dV_Total_Pareto,Cl_Pareto,4);
% [f_Cl, delta_Cl] = polyconf(fit_Cl,dV_Range,S_Cl, ...
%                             'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_Cl,'-b');
% hold on; plot(dV_Range,f_Cl + delta_Cl,':b');
% hold on; plot(dV_Range,f_Cl - delta_Cl,':b');

ylim([0.0 4.0]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Lift Coefficient');

% legend('Pareto Data','Polynomial Fit, Degree: 4', ...
%         '95% Confidence Bounds','location','SouthEast');
legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Total Delta-V (km/s)
subplot(2,3,6); box on; grid off;
plot(dV_Total_Pareto,Perig_Pareto,'r. ');

% %Polynomial fit
% [fit_Perig,S_Perig] = polyfit(dV_Total_Pareto,Perig_Pareto,2);
% [f_Perig, delta_Perig] = polyconf(fit_Perig,dV_Range,S_Perig, ...
%                                  'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_Perig,'-b');
% hold on; plot(dV_Range,f_Perig + delta_Perig,':b');
% hold on; plot(dV_Range,f_Perig - delta_Perig,':b');

ylim([75 115]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Perigee Altitude, km');

% legend('Pareto Data','Polynomial Fit, Degree: 2', ...
%         '95% Confidence Bounds','location','SouthEast');
legend('Boundary Data','location','NorthEast');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Total Delta-V (km/s)
% figure; box on; grid off;
% plot(dV_Total_Pareto,BC_Pareto,'r. ');

```

```

%
% %Polynomial fit
% [fit_BC,S_BC] = polyfit(dV_Total_Pareto,BC_Pareto,4);
% [f_BC, delta_BC] = polyconf(fit_BC,dV_Range,S_BC, ...
%                               'simopt','on','predopt','curve');
% hold on; plot(dV_Range,f_BC,'-b');
% hold on; plot(dV_Range,f_BC + delta_BC,':b');
% hold on; plot(dV_Range,f_BC - delta_BC,':b');
%
% % ylim([-0.1 0.1]);
% xlabel('Total \it\DeltaV\rm, km/s');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','Polynomial Fit, Degree: 4', ...
%        '95% Confidence Bounds','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

```

ParetoBoundary_6Factors.m

```

close all; clear all; clc;

Pareto_Choice = 1;
%1 = MAX Delta-Inclination, MIN Delta-V
%2 = MAX Delta-Inclination, MAX Re-Circularization Altitude
%3 = MIN Delta-V,                MAX Re-Circularization Altitude

Pareto_Intersect_Choice = 2;
%1 = Identifies, plots, and saves common Pareto optimal points
%2 = Converse of Choice #1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Pareto_Choice == 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MIN Delta-V
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 8; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_dV = 0; %Constraint for minimum delta-V (km/s)
obj_x = 7; %Column number of x-axis objective (from reduced matrix)
obj_y = 6; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

```



```

%Loads experiments and observations
load DOEMatrix_729_0_120;

%Matrix of experiments and observations
IN = DOEMatrix_729_0_120(:,1:end);
split = length(IN(:,1));
total = length(IN(:,1)); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_dV);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_dV)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\Delta V\rm, km/s');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
p = scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
            J_filt_tot(K,obj_y)./BaselineCost','y', ...
            'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
            'MarkerEdgeColor','k','LineWidth',1,'HandleVisibility','off');
legend(p,{ 'Pareto Optimal Points'}, 'location','NorthEast');

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);

```

```

J_sorted1 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted1 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto_xJ = [x_sorted1,J_sorted1];

%Saves Pareto points to .MAT file
savefile = 'ParetoPoints_3125.mat';
save(savefile,'x_sorted1','J_sorted1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto      = x_sorted1(:,2); %Mass (kg)
S_Pareto         = x_sorted1(:,3); %Planform area (m^2)
Cd_Pareto        = x_sorted1(:,4); %Drag coefficient
Cl_Pareto        = x_sorted1(:,5); %Lift coefficient
Perig_Pareto     = x_sorted1(:,6); %Perigee altitude (km)
InitAlt_Pareto   = x_sorted1(:,7); %Initial altitude (km)
Bank_Pareto      = x_sorted1(:,8); %Bank angle (deg)
BC_Pareto        = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted1(:,2); %Re-circularization altitude (km)
dV_Total_Pareto  = J_sorted1(:,6); %Total delta-V (km/s)
MaxIncl_Pareto   = J_sorted1(:,7); %Maximum inclination (deg)
dIncl_Pareto     = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range      = [0:0.1:16]'; %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'r. ');
ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r. ');
ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aerodynamic Coefficients v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
[AX,Cd1,C12] = plotyy(dIncl_Pareto,Cd_Pareto,dIncl_Pareto,C1_Pareto,'plot');
xlabel('Maximum Inclination Change, deg');
set(Cd1,'linestyle','none','Marker','.');
set(C12,'linestyle','none','Marker','.');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(C12,'color','blue');
set(AX,{ 'ycolor' },{ 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r.');
ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Bank Angle (deg) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,Bank_Pareto,'r.');
ylim([-120 0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Bank Angle, deg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
% ylim([-0.1 0.1]);
% xlabel('Maximum Inclination Change, deg');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

elseif Pareto_Choice == 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MAX Re-Circ. Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 8; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_alt = 130; %Constraint for re-circularization altitude (km)
obj_x = 7; %Column number of x-axis objective (from reduced matrix)
obj_y = 2; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_729_0_120;

%Matrix of experiments and observations
IN = DOEMatrix_729_0_120(:,1:end);
split = length(IN);
total = length(IN); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot      = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
p = scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
            J_filt_tot(K,obj_y)./BaselineCost', 'y', ...
            'SizeData',5^2, 'MarkerFaceColor', 'r', 'Marker', 'o', ...
            'MarkerEdgeColor', 'k', 'LineWidth',1, 'HandleVisibility', 'off');

legend(p,{'Pareto Optimal Points'},'location','NorthEast');

if Pareto_Intersect_Choice == 1
    %Loads and plots Pareto points from {max(delta-i),min(delta-V)} analysis
    load ParetoPoints_3125.mat;
    Pareto1 = [x_sorted1,J_sorted1];
    hold on; plot((J_sorted1(:,7) - InitIncl), ...
                  J_sorted1(:,2), 'go', 'LineWidth',2);
end

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto      = J_filt_tot(K,:);
[B IX]        = sort(J_pareto,1);
J_sorted2     = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted2     = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto2       = [x_sorted2,J_sorted2];

if Pareto_Intersect_Choice == 1
    %Common Pareto points between
    %{max(delta-i),min(delta-V)} and {max(delta-i),max(h_recirc)}
    Pareto_Intersect_12 = intersect(Pareto1,Pareto2,'rows');

    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_12.mat';
    save(savefile,'Pareto_Intersect_12');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

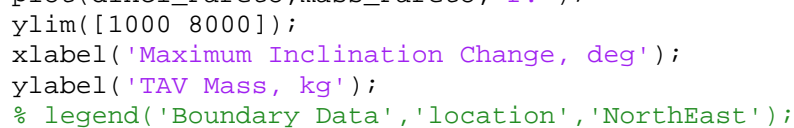
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto      = x_sorted2(:,2); %Mass (kg)
S_Pareto         = x_sorted2(:,3); %Planform area (m^2)
Cd_Pareto        = x_sorted2(:,4); %Drag coefficient
Cl_Pareto        = x_sorted2(:,5); %Lift coefficient
Perig_Pareto     = x_sorted2(:,6); %Perigee altitude (km)
InitAlt_Pareto   = x_sorted2(:,7); %Initial altitude (km)
Bank_Pareto      = x_sorted2(:,8); %Bank angle (deg)
BC_Pareto        = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted2(:,2); %Re-circularization altitude (km)
dV_Total_Pareto  = J_sorted2(:,6); %Total delta-V (km/s)
MaxIncl_Pareto   = J_sorted2(:,7); %Maximum inclination (deg)
dIncl_Pareto     = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range      = [0:0.1:16]';      %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'r.');
```

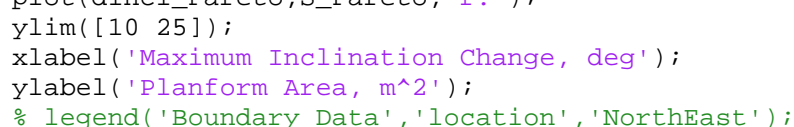


```

ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r.');
```

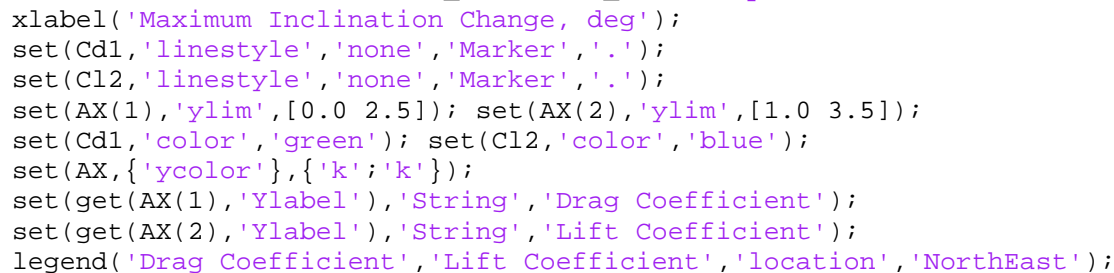


```

ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aerodynamic Coefficients v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
[AX,Cd1,Cl2] = plotyy(dIncl_Pareto,Cd_Pareto, ...
                    dIncl_Pareto,Cl_Pareto,'plot');
```

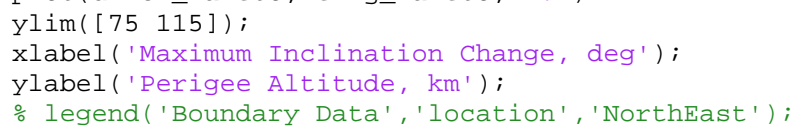


```

xlabel('Maximum Inclination Change, deg');
set(Cd1,'linestyle','none','Marker','.');
set(Cl2,'linestyle','none','Marker','.');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(Cl2,'color','blue');
set(AX,{ 'ycolor' }, { 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r.');
```

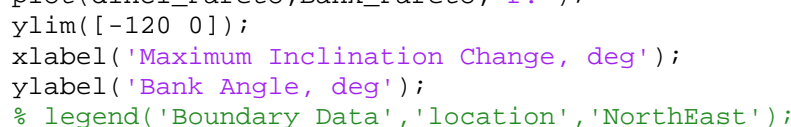


```

ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Bank Angle (deg) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,Bank_Pareto,'r.');
```

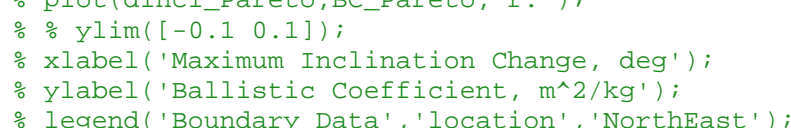


```

ylim([-120 0]);
xlabel('Maximum Inclination Change, deg');
ylabel('Bank Angle, deg');
% legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
```



```

% % ylim([-0.1 0.1]);
% xlabel('Maximum Inclination Change, deg');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','location','NorthEast');
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif Pareto_Choice == 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MIN Delta-V, MAX Re-Circularization Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars     = 8;      %Number of columns for experiment number and factors
min_dV    = 0;      %Constraint for minimum inclination (deg)
min_alt    = 130;    %Constraint for re-circularization altitude (km)
obj_x     = 6;      %Column number of x-axis objective (from reduced matrix)
obj_y     = 2;      %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1;   %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_729_0_120;

%Matrix of experiments and observations
IN      = DOEMatrix_729_0_120(:,1:end);
split   = length(IN);
total   = length(IN); %Length of input matrix (number of rows)

```

```

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:)      = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_dV & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_dV & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x))',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Total \it\Delta V\rm, km/s');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)]==1));
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)]==1));

%Plotting of Pareto front
hold on;
p = scatter((J_filt_tot(K,obj_x))', ...
            J_filt_tot(K,obj_y)./BaselineCost','y', ...
            'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
            'MarkerEdgeColor','k','LineWidth',0.1,'HandleVisibility','off');

legend(p,{'Pareto Optimal Points'},'location','NorthWest');

if Pareto_Intersect_Choice == 1
    %Loads/plots common Pareto points between {max(delta-i),min(delta-V)}
    %and {max(delta-i),max(h_recirc)}
    load Pareto_Intersect_12.mat;
    hold on; p = plot(Pareto_Intersect_12(:,14), ...
                    Pareto_Intersect_12(:,10),'go','LineWidth',2);
end

```



```

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted3 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted3 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto3 = [x_sorted3,J_sorted3];

if Pareto_Intersect_Choice == 1
    %Common Pareto points between {max(delta-i),min(delta-V)},
    %{max(delta-i),max(h_recirc)}, and {min(delta-V),max(h_recirc)}
    Pareto_Intersect_123 = intersect(Pareto_Intersect_12,Pareto3,'rows');
    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_123.mat';
    save(savefile,'Pareto_Intersect_123');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted3(:,2); %Mass (kg)
S_Pareto = x_sorted3(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted3(:,4); %Drag coefficient
Cl_Pareto = x_sorted3(:,5); %Lift coefficient
Perig_Pareto = x_sorted3(:,6); %Perigee altitude (km)
InitAlt_Pareto = x_sorted3(:,7); %Initial altitude (km)
Bank_Pareto = x_sorted3(:,8); %Bank angle (deg)
BC_Pareto = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted3(:,2); %Re-circularization altitude (km)
dV_Total_Pareto = J_sorted3(:,6); %Total delta-V (km/s)
MaxIncl_Pareto = J_sorted3(:,7); %Maximum inclination (deg)
dIncl_Pareto = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range = [0:0.1:16]'; %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Total Delta-V (km/s)
subplot(2,3,2); box on; grid off;
plot(dV_Total_Pareto,mass_Pareto,'r. ');
ylim([1000 8000]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Total Delta-V (km/s)
subplot(2,3,3); box on; grid off;
plot(dV_Total_Pareto,S_Pareto,'r. ');
ylim([10 25]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aerodynamic Coefficients v. Total Delta-V (km/s)
subplot(2,3,4); box on; grid off;
[AX,Cd1,C12] = plotyy(dV_Total_Pareto,Cd_Pareto, ...
                    dV_Total_Pareto,C1_Pareto,'plot');
xlabel('Total \it\DeltaV\rm, km/s');
set(Cd1,'linestyle','none','Marker','. ');
set(C12,'linestyle','none','Marker','. ');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(C12,'color','blue');
set(AX,{ 'ycolor' },{ 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Total Delta-V (km/s)
subplot(2,3,5); box on; grid off;
plot(dV_Total_Pareto,Perig_Pareto,'r. ');
ylim([75 115]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Bank Angle (deg) v. Total Delta-V (km/s)
subplot(2,3,6); box on; grid off;
plot(dV_Total_Pareto,Bank_Pareto,'r. ');
ylim([-120 0]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Bank Angle, deg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

```

ParetoBoundary_InitAlt.m

```
close all; clear all; clc;

Pareto_Choice = 1;
%1 = MAX Delta-Inclination, MIN Delta-V
%2 = MAX Delta-Inclination, MAX Re-Circularization Altitude
%3 = MIN Delta-V, MAX Re-Circularization Altitude

Pareto_Intersect_Choice = 2;
%1 = Identifies, plots, and saves common Pareto optimal points
%2 = Converse of Choice #1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Pareto_Choice == 1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MIN Delta-V
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 6; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_dV = 0; %Constraint for minimum delta-V (km/s)
obj_x = 9; %Column number of x-axis objective (from reduced matrix)
obj_y = 1; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_300_1000;

%Matrix of experiments and observations
IN = DOEMatrix_3125_300_1000(:,1:end);
split = length(IN(:,1));
total = length(IN(:,1)); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_dV);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_dV)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);
```

```

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
ylim([100 1100]);
xlabel('Maximum Inclination Change, deg');
ylabel('Initial Altitude, km');
% ylabel('Total \it\DeltaV\rm, km/s');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
% p = scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
%             J_filt_tot(K,obj_y)./BaselineCost','y', ...
%             'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
%             'MarkerEdgeColor','k','LineWidth',1,'HandleVisibility','off');
% legend(p,{'Pareto Optimal Points'},'location','SouthEast');

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted1 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted1 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto_xJ = [x_sorted1,J_sorted1];

%Saves Pareto points to .MAT file
savefile = 'ParetoPoints_3125.mat';
save(savefile,'x_sorted1','J_sorted1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted1(:,2); %Mass (kg)
S_Pareto = x_sorted1(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted1(:,4); %Drag coefficient
Cl_Pareto = x_sorted1(:,5); %Lift coefficient
Perig_Pareto = x_sorted1(:,6); %Perigee altitude (km)
InitAlt_Pareto = x_sorted1(:,7); %Initial altitude (km)

```

```

Bank_Pareto      = x_sorted1(:,8); %Bank angle (deg)
BC_Pareto       = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted1(:,2); %Re-circularization altitude (km)
dV_Total_Pareto  = J_sorted1(:,6); %Total delta-V (km/s)
MaxIncl_Pareto   = J_sorted1(:,7); %Maximum inclination (deg)
dIncl_Pareto     = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range      = [0:0.1:16]';      %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'r. ');
ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r. ');
ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');

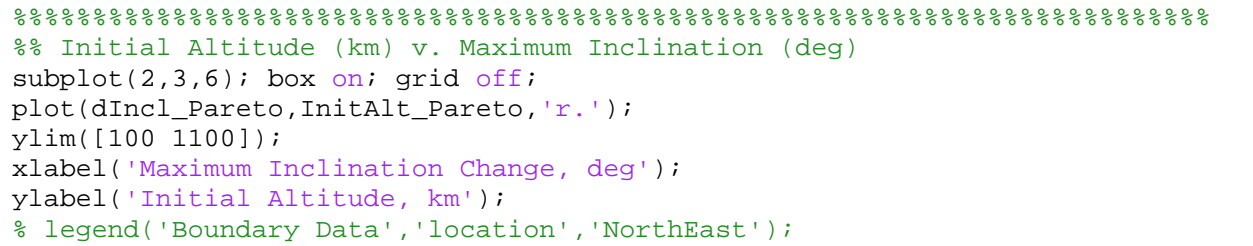
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Aerodynamic Coefficients v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
[AX,Cd1,C12] = plotyy(dIncl_Pareto,Cd_Pareto, ...
                     dIncl_Pareto,C1_Pareto,'plot');
xlabel('Maximum Inclination Change, deg');
set(Cd1,'linestyle','none','Marker','. ');
set(C12,'linestyle','none','Marker','. ');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(C12,'color','blue');
set(AX,{ 'ycolor' }, { 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r. ');
ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,InitAlt_Pareto,'r.');
```



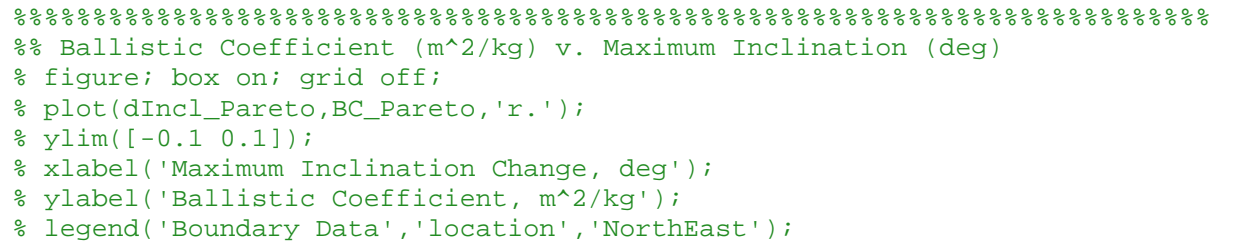
```

ylim([100 1100]);
xlabel('Maximum Inclination Change, deg');
ylabel('Initial Altitude, km');
% legend('Boundary Data','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
```



```

% ylim([-0.1 0.1]);
% xlabel('Maximum Inclination Change, deg');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','location','NorthEast');
```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```

elseif Pareto_Choice == 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MAX Re-Circ. Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 8; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_alt = 130; %Constraint for re-circularization altitude (km)
obj_x = 7; %Column number of x-axis objective (from reduced matrix)
obj_y = 2; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_300_1000;

%Matrix of experiments and observations
IN = DOEMatrix_3125_300_1000(:,1:end);
split = length(IN);
total = length(IN); %Length of input matrix (number of rows)

%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:) = IN(i,nvars+1:size(IN,2)); %Observations
end
```

```

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_incl & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_incl & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x)-InitIncl)',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([-J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
        J_filt_tot(K,obj_y)./BaselineCost','y', ...
        'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
        'MarkerEdgeColor','k','LineWidth',1,'HandleVisibility','off');

if Pareto_Intersect_Choice == 1
    %Loads and plots Pareto points from {max(delta-i),min(delta-V)} analysis
    load ParetoPoints_3125.mat;
    Pareto1 = [x_sorted1,J_sorted1];
    hold on; plot((J_sorted1(:,7) - InitIncl), ...
        J_sorted1(:,2),'go','LineWidth',2);
end

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted2 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front

x_sorted2 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto2 = [x_sorted2,J_sorted2];

```

```

if Pareto_Intersect_Choice == 1
    %Common Pareto points between
    %{max(delta-i),min(delta-V)} and {max(delta-i),max(h_recirc)}
    Pareto_Intersect_12 = intersect(Pareto1,Pareto2,'rows');

    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_12.mat';
    save(savefile,'Pareto_Intersect_12');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto      = x_sorted2(:,2); %Mass (kg)
S_Pareto         = x_sorted2(:,3); %Planform area (m^2)
Cd_Pareto        = x_sorted2(:,4); %Drag coefficient
Cl_Pareto        = x_sorted2(:,5); %Lift coefficient
Perig_Pareto     = x_sorted2(:,6); %Perigee altitude (km)
InitAlt_Pareto   = x_sorted2(:,7); %Initial altitude (km)
Bank_Pareto      = x_sorted2(:,8); %Bank angle (deg)
BC_Pareto        = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted2(:,2); %Re-circularization altitude (km)
dV_Total_Pareto  = J_sorted2(:,6); %Total delta-V (km/s)
MaxIncl_Pareto   = J_sorted2(:,7); %Maximum inclination (deg)
dIncl_Pareto     = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range      = [0:0.1:16]';      %Delta-inclination angle range (deg)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Maximum Inclination (deg)
subplot(2,3,2); box on; grid off;
plot(dIncl_Pareto,mass_Pareto,'r. ');
ylim([1000 8000]);
xlabel('Maximum Inclination Change, deg');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Maximum Inclination (deg)
subplot(2,3,3); box on; grid off;
plot(dIncl_Pareto,S_Pareto,'r. ');
ylim([10 25]);
xlabel('Maximum Inclination Change, deg');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aerodynamic Coefficients v. Maximum Inclination (deg)
subplot(2,3,4); box on; grid off;
[AX,Cd1,C12] = plotyy(dIncl_Pareto,Cd_Pareto, ...
                    dIncl_Pareto,C1_Pareto,'plot');
xlabel('Maximum Inclination Change, deg');
set(Cd1,'linestyle','none','Marker','.');
set(C12,'linestyle','none','Marker','.');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(C12,'color','blue');
set(AX,{ 'ycolor' }, { 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,5); box on; grid off;
plot(dIncl_Pareto,Perig_Pareto,'r.');
ylim([75 115]);
xlabel('Maximum Inclination Change, deg');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Altitude (km) v. Maximum Inclination (deg)
subplot(2,3,6); box on; grid off;
plot(dIncl_Pareto,InitAlt_Pareto,'r.');
ylim([100 1100]);
xlabel('Maximum Inclination Change, deg');
ylabel('Initial Altitude, km');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Maximum Inclination (deg)
% figure; box on; grid off;
% plot(dIncl_Pareto,BC_Pareto,'r.');
% % ylim([-0.1 0.1]);
% xlabel('Maximum Inclination Change, deg');
% ylabel('Ballistic Coefficient, m^2/kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

elseif Pareto_Choice == 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pareto Optimization: MIN Delta-V, MAX Re-Circularization Altitude
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars     = 8;      %Number of columns for experiment number and factors
min_dV    = 0;      %Constraint for minimum inclination (deg)
min_alt   = 130;    %Constraint for re-circularization altitude (km)
obj_x     = 6;      %Column number of x-axis objective (from reduced matrix)
obj_y     = 2;      %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1;   %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_300_1000;

%Matrix of experiments and observations
IN = DOEMatrix_3125_300_1000(:,1:end);
split = length(IN);
total = length(IN); %Length of input matrix (number of rows)
%Creation of reduced factor and observation matrices
for i = 1:size(IN,1)
    x_star(i,:) = IN(i,1:nvars); %Factors (w/ experiment number)
    J(i,:)      = IN(i,nvars+1:size(IN,2)); %Observations
end

%Determination of observations which satisfy constraints
I = find(J(1:split,obj_x) >= min_dV & ...
        J(1:split,obj_y) >= min_alt);
Z = (find(J(split+1:total,obj_x) >= min_dV & ...
        J(split+1:total,obj_y) >= min_alt)+split);
J_filt = J(I,:);
J_filt1 = J(Z,:);

%Factors associated with observations which satisfy constraints
x_star_filt = x_star(I,:);
x_star_filt1 = x_star(Z,:);

%Plotting of design space
subplot(2,3,1);
scatter((J_filt(:,obj_x))',J_filt(:,obj_y)./BaselineCost','b', ...
        'SizeData',5^2,'MarkerFaceColor','k','Marker','.');
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Re-Circularization Altitude, km');
hold on; box on; grid off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt; J_filt1];
x_star_filt_tot = [x_star_filt; x_star_filt1];

K = find(paretofront([J_filt_tot(:,obj_x) -J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

```

```

%Plotting of Pareto front
hold on;
scatter((J_filt_tot(K,obj_x))', ...
        J_filt_tot(K,obj_y)./BaselineCost','y', ...
        'SizeData',5^2,'MarkerFaceColor','r','Marker','o', ...
        'MarkerEdgeColor','k','LineWidth',0.5,'HandleVisibility','off');

if Pareto_Intersect_Choice == 1
    %Loads/plots common Pareto points between {max(delta-i),min(delta-V)}
    %and {max(delta-i),max(h_recirc)}
    load Pareto_Intersect_12.mat;
    hold on; plot(Pareto_Intersect_12(:,14), ...
        Pareto_Intersect_12(:,10),'go','LineWidth',2);
end

R = find(K > size(J_filt,1));
x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);
[B IX] = sort(J_pareto,1);
J_sorted3 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted3 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto3 = [x_sorted3,J_sorted3];

if Pareto_Intersect_Choice == 1
    %Common Pareto points between {max(delta-i),min(delta-V)},
    %{max(delta-i),max(h_recirc)}, and {min(delta-V),max(h_recirc)}
    Pareto_Intersect_123 = intersect(Pareto_Intersect_12,Pareto3,'rows');
    %Saves common Pareto points to .MAT file
    savefile = 'Pareto_Intersect_123.mat';
    save(savefile,'Pareto_Intersect_123');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'ScreenSize')); %Automatically maximizes plot window

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Factor Analysis of Pareto Front
%Factors
mass_Pareto = x_sorted3(:,2); %Mass (kg)
S_Pareto = x_sorted3(:,3); %Planform area (m^2)
Cd_Pareto = x_sorted3(:,4); %Drag coefficient
Cl_Pareto = x_sorted3(:,5); %Lift coefficient
Perig_Pareto = x_sorted3(:,6); %Perigee altitude (km)
InitAlt_Pareto = x_sorted3(:,7); %Initial altitude (km)
Bank_Pareto = x_sorted3(:,8); %Bank angle (deg)
BC_Pareto = ((Cd_Pareto.*S_Pareto)./(2*mass_Pareto)); %Ballistic coeff.

%Observations
RecircAlt_Pareto = J_sorted3(:,2); %Re-circularization altitude (km)
dV_Total_Pareto = J_sorted3(:,6); %Total delta-V (km/s)
MaxIncl_Pareto = J_sorted3(:,7); %Maximum inclination (deg)
dIncl_Pareto = MaxIncl_Pareto - InitIncl; %Max. inclination change (deg)
dIncl_Range = [0:0.1:16]'; %Delta-inclination angle range (deg)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Mass (kg) v. Total Delta-V (km/s)
subplot(2,3,2); box on; grid off;
plot(dV_Total_Pareto,mass_Pareto,'r. ');
ylim([1000 8000]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('TAV Mass, kg');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Planform Area (m^2) v. Total Delta-V (km/s)
subplot(2,3,3); box on; grid off;
plot(dV_Total_Pareto,S_Pareto,'r. ');
ylim([10 25]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Planform Area, m^2');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aerodynamic Coefficients v. Total Delta-V (km/s)
subplot(2,3,4); box on; grid off;
[AX,Cd1,C12] = plotyy(dV_Total_Pareto,Cd_Pareto, ...
                    dV_Total_Pareto,C1_Pareto,'plot');
xlabel('Total \it\DeltaV\rm, km/s');
set(Cd1,'linestyle','none','Marker','. ');
set(C12,'linestyle','none','Marker','. ');
set(AX(1),'ylim',[0.0 2.5]); set(AX(2),'ylim',[1.0 3.5]);
set(Cd1,'color','green'); set(C12,'color','blue');
set(AX,{ 'ycolor' }, { 'k'; 'k' });
set(get(AX(1),'Ylabel'),'String','Drag Coefficient');
set(get(AX(2),'Ylabel'),'String','Lift Coefficient');
legend('Drag Coefficient','Lift Coefficient','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perigee Altitude (km) v. Total Delta-V (km/s)
subplot(2,3,5); box on; grid off;
plot(dV_Total_Pareto,Perig_Pareto,'r. ');
ylim([75 115]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Perigee Altitude, km');
% legend('Boundary Data','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Altitude (km) v. Total Delta-V (km/s)
subplot(2,3,6); box on; grid off;
plot(dV_Total_Pareto,InitAlt_Pareto,'r. ');
ylim([100 1100]);
xlabel('Total \it\DeltaV\rm, km/s');
ylabel('Initial Altitude, km');
% legend('Boundary Data','location','NorthEast');

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Ballistic Coefficient (m^2/kg) v. Total Delta-V (km/s)
% figure; box on; grid off;
% plot(dV_Total_Pareto,BC_Pareto,'r.');
```

% % ylim([-0.1 0.1]);

% xlabel('Total \it\Delta V\rm, km/s');

% ylabel('Ballistic Coefficient, m^2/kg');

% legend('Boundary Data','location','NorthEast');

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

```

ParetoDOE.m

```

close all; clear all; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pareto Optimization: MAX Delta-Inclination, MIN Delta-V
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initial Conditions for Pareto Analysis
InitIncl = 37.843; %Initial inclination (deg)
nvars = 6; %Number of columns for experiment number and factors
min_incl = InitIncl; %Constraint for minimum inclination (deg)
min_dV = 0; %Constraint for minimum delta-V (km/s)
obj_x = 9; %Column number of x-axis objective (from reduced matrix)
obj_y = 8; %Column number of y-axis objective (from reduced matrix)
BaselineCost = 1; %Value to normalize y-axis objective

%Loads experiments and observations
load DOEMatrix_3125_5_79; %Initial DOE Matrix
load ParetoMatrix_3125_5_79; %Pareto Front DOE Matrix
load DOEOutliers_3125_5_79; %Outlier DOE Matrix

%% Matrix of Experiments and Observations
%Initial DOE Matrix
IN_1 = DOEMatrix_3125_5_79(:,1:end);
split_1 = length(IN_1(:,1));
total_1 = length(IN_1(:,1)); %Length of input matrix (number of rows)

%Pareto Front DOE Matrix
IN_2 = ParetoMatrix_3125_5_79(:,1:end);
split_2 = length(IN_2(:,1));
total_2 = length(IN_2(:,1)); %Length of input matrix (number of rows)

```

```

%Outlier DOE Matrix
IN_3 = DOEOutliers_3125_5_79(:,1:end);
split_3 = length(IN_3(:,1));
total_3 = length(IN_3(:,1)); %Length of input matrix (number of rows)

%% Creation of Reduced Factor and Observation Matrices
%Initial DOE Matrix
for ii = 1:size(IN_1,1)
    x_star_1(ii,:) = IN_1(ii,1:nvars); %Factors
    J_1(ii,:) = IN_1(ii,nvars+1:size(IN_1,2)); %Observations
end

%Pareto Front DOE Matrix
for jj = 1:size(IN_2,1)
    x_star_2(jj,:) = IN_2(jj,1:nvars); %Factors
    J_2(jj,:) = IN_2(jj,nvars+1:size(IN_2,2)); %Observations
end

%Outlier DOE Matrix
for kk = 1:size(IN_3,1)
    x_star_3(kk,:) = IN_3(kk,1:nvars); %Factors
    J_3(kk,:) = IN_3(kk,nvars+1:size(IN_3,2)); %Observations
end

%% Determination of Observations which Satisfy Constraints
%Initial DOE Matrix
I_1 = find(J_1(1:split_1,obj_x) >= min_incl & ...
           J_1(1:split_1,obj_y) >= min_dV);
Z_1 = (find(J_1(split_1+1:total_1,obj_x) >= min_incl & ...
           J_1(split_1+1:total_1,obj_y) >= min_dV)+split_1);
J_filt_1 = J_1(I_1,:);
J_filt1_1 = J_1(Z_1,:);

%Pareto Front DOE Matrix
I_2 = find(J_2(1:split_2,obj_x) >= min_incl & ...
           J_2(1:split_2,obj_y) >= min_dV);
Z_2 = (find(J_2(split_2+1:total_2,obj_x) >= min_incl & ...
           J_2(split_2+1:total_2,obj_y) >= min_dV)+split_2);
J_filt_2 = J_2(I_2,:);
J_filt1_2 = J_2(Z_2,:);

%Outlier DOE Matrix
I_3 = find(J_3(1:split_3,obj_x) >= min_incl & ...
           J_3(1:split_3,obj_y) >= min_dV);
Z_3 = (find(J_3(split_3+1:total_3,obj_x) >= min_incl & ...
           J_3(split_3+1:total_3,obj_y) >= min_dV)+split_3);
J_filt_3 = J_3(I_3,:);
J_filt1_3 = J_3(Z_3,:);

%% Factors Associated with Observations which Satisfy Constraints
%Initial DOE Matrix
x_star_filt_1 = x_star_1(I_1,:);
x_star_filt1_1 = x_star_1(Z_1,:);

```

```

%Pareto Front DOE Matrix
x_star_filt_2 = x_star_2(I_2,:);
x_star_filt1_2 = x_star_2(Z_2,:);

%Outlier DOE Matrix
x_star_filt_3 = x_star_3(I_3,:);
x_star_filt1_3 = x_star_3(Z_3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting of Design Space
figure; %Initial DOE Matrix
scatter((J_filt_1(:,obj_x)-InitIncl)',J_filt_1(:,obj_y)./ ...
    BaselineCost','b','SizeData',6^2,'MarkerFaceColor','k','Marker','.');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\DeltaV\rm, km/s');

hold on; %Pareto Front DOE Matrix
scatter((J_filt_2(:,obj_x)-InitIncl)',J_filt_2(:,obj_y)./ ...
    BaselineCost','g','SizeData',6^2.5,'MarkerFaceColor','k','Marker','x');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\DeltaV\rm, km/s');
hold on;

hold on; %Outlier DOE Matrix
scatter((J_filt_3(:,obj_x)-InitIncl)',J_filt_3(:,obj_y)./ ...
    BaselineCost','r','SizeData',6^2,'MarkerFaceColor','w','Marker','o');
xlabel('Maximum Inclination Change, deg');
ylabel('Total \it\DeltaV\rm, km/s');
hold on; box on; grid off;

legend('Initial DOE Campaign','DOE for Pareto Front', ...
    'DOE for Outlier Points','location','NorthEast');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Data Filtering and Pareto Analysis
J_filt_tot = [J_filt_1; J_filt1_1; ...
    J_filt_2; J_filt1_2; ...
    J_filt_3; J_filt1_3];
% J_filt_tot = [J_filt_1; J_filt1_1; ...
% J_filt_2; J_filt1_2];

x_star_filt_tot = [x_star_filt_1; x_star_filt1_1; ...
    x_star_filt_2; x_star_filt1_2; ...
    x_star_filt_3; x_star_filt1_3];
% x_star_filt_tot = [x_star_filt_1; x_star_filt1_1; ...
% x_star_filt_2; x_star_filt1_2];

K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);
% K = find(paretofront([-J_filt_tot(:,obj_x) J_filt_tot(:,obj_y)])==1);

%Plotting of Pareto front
hold on;

```

```

scatter((J_filt_tot(K,obj_x) - InitIncl)', ...
        J_filt_tot(K,obj_y)./BaselineCost','y', ...
        'SizeData',10^2.5,'Marker','s', ...
        'MarkerEdgeColor','k','LineWidth',1.5,'HandleVisibility','off');

x_star_pareto = x_star_filt_tot(K,:);
J_pareto = J_filt_tot(K,:);

[B IX] = sort(J_pareto,1);
J_sorted1 = J_pareto([IX(:,obj_x)'],:); %Observations for Pareto front
x_sorted1 = x_star_pareto([IX(:,obj_x)'],:); %Factors for Pareto front
Pareto_xJ = [x_sorted1,J_sorted1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Miscellaneous Plotting Commands
set(gcf,'Color','w'); %Sets overall figure background color to 'white'
set(gcf,'Position',get(0,'Screensize')); %Automatically maximizes plot window

```

paretofront.m

```

function [] = paretofront(varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Use: front = paretofront(M)
%
% Returns the logical Pareto Front of a set of points.
%
% Author/Date      : Cao, Yi/Cranfield University/2007
% Modified by     : Bettinger, Robert AFIT/ENY/2013
%
% Example:
%   Find the Pareto Front of a set of 3D random points:
%       X = rand(100,3);
%       front = paretofront(X);
%       hold on;
%       plot3(X(:,1),X(:,2),X(:,3),'.');
%       plot3(X(front, 1) , X(front, 2) , X(front, 3) , 'r.');
```

```

%       hold off; grid on;
%       view(-37.5, 30)
%       xlabel('X_1'); ylabel('X_2'); zlabel('X_3');
%       title('Pareto Front of a set of random points in 3D');
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

error('mex file absent, type ''mex paretofront.c'' to compile');
```



```
#include <math.h>
#include "mex.h"

/*
    paretomember returns the logical Pareto membership of a set of points.

    Synopsis      : front = paretofront(objMat)

    Author/Date   : Cao, Yi/Cranfield University/2007

    Modified by   : Bettinger, Robert AFIT/ENY/2013

    mex paretofront.c
*/

void paretofront(bool * front, double * M, unsigned int row, unsigned int
col);

void mexFunction( int nlhs, mxArray *plhs[] , int nrhs, const mxArray *prhs[]
)
{
    bool * front;
    double * M;
    unsigned int row, col;
    const int *dims;

    if(nrhs == 0 || nlhs > 1)
    {
        printf("\nsynopsis:   front = paretofront(X)");
        plhs[0] = mxCreateDoubleMatrix(0 , 0 , mxREAL);
        return;
    }

    M = mxGetPr(prhs[0]);
    dims = mxGetDimensions(prhs[0]);
    row = dims[0];
    col = dims[1];

    /* ----- Output ----- */

    plhs[0] = mxCreateLogicalMatrix (row , 1);
    front = (bool *) mxGetPr(plhs[0]);

    /* --- Main Call --- */
    paretofront(front, M, row, col);
}

void paretofront(bool * front, double * M, unsigned int row, unsigned int
col)
{
```

```

unsigned int t,s,i,j,j1,j2;
bool *checklist, coldominatedflag;

checklist = (bool *)mxMalloc(row*sizeof(bool));
for(t = 0; t<row; t++) checklist[t] = true;
for(s = 0; s<row; s++) {
    t=s;
    if (!checklist[t]) continue;
    checklist[t]=false;
    coldominatedflag=true;
    for(i=t+1;i<row;i++) {
        if (!checklist[i]) continue;
        checklist[i]=false;
        for (j=0,j1=i,j2=t;j<col;j++,j1+=row,j2+=row) {
            if (M[j1] < M[j2]) {
                checklist[i]=true;
                break;
            }
        }
        if (!checklist[i]) continue;
        coldominatedflag=false;
        for (j=0,j1=i,j2=t;j<col;j++,j1+=row,j2+=row) {
            if (M[j1] > M[j2]) {
                coldominatedflag=true;
                break;
            }
        }
        if (!coldominatedflag) { //Swap active index continue checking
            front[t]=false;
            checklist[i]=false;
            coldominatedflag=true;
            t=i;
        }
    }
    front[t]=coldominatedflag;
    if (t>s) {
        for (i=s+1; i<t; i++) {
            if (!checklist[i]) continue;
            checklist[i]=false;
            for (j=0,j1=i,j2=t;j<col;j++,j1+=row,j2+=row) {
                if (M[j1] < M[j2]) {
                    checklist[i]=true;
                    break;
                }
            }
        }
    }
}
mxFree(checklist);
}

```

References

- Agte, Jeremy S. "Multistate Analysis and Design: Case Studies in Aerospace Design and Long Endurance Systems." Ph.D Dissertation. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology (MIT), Cambridge, MA, 2011.
- Agte, Jeremy S., Nicholas Borer, and Olivier de Weck. "Design of Long Endurance Systems with Inherent Robustness to Partial Failures during Operations," *Journal of Mechanical Design* 134, no. 10: 100903-100918 (2012).
- Akins, Keith, Liam Healy, Shannon Coffey, and Mike Picone. "Comparison of MSIS and Jacchia Atmospheric Density Models for Orbit Determination and Propagation." Paper presented at the *13th AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico, 9-13 February 2003.
- Allen, H. J. and A. J. Eggers, Jr. "A Study of the Motion and Aerodynamic Heating of Ballistic Missiles Entering the Earth's Atmosphere at High Supersonic Speeds," *NACA TR 1381*. Moffett Field, CA: AMES Aeronautical Laboratory, 1958.
- Allen, Julie D. and Unicode Consortium, eds. *The Unicode Standard: Version 5.0*. Reading, MA: Addison-Wesley, 2007.
- Anderson Jr., John D. *Hypersonic and High-Temperature Gas Dynamics*, Second Edition. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2006.
- Arora, J. S. *Introduction to Optimum Design*, Third Edition. Waltham, MA: Academic Press, 2012.
- Barbeau, Edward J. *Polynomials*. New York, NY: Springer-Verlag New York, Inc., 1989.
- Barrentine, Larry B. *An Introduction to Design of Experiments: A Simplified Approach*. Milwaukee, WI: Quality Press, 1999.
- Bate, Roger R., Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics*. New York, NY: Dover Publications, Inc., 1971.
- Battin, Richard H. and Robin M. Vaughn. "An Elegant Lambert Algorithm," *Journal of Spacecraft and Rockets* 7, no. 6: 662-670 (1984).
- Bedford, Anthony and Wallace Fowler. *Engineering Mechanics: Dynamics*, Fourth Edition. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.
- Betin, Pierre. "Reflections on the Spaceplane," *Space Policy* 7, no. 2: 137-145 (1991).

- Bettinger, Robert A. and Jonathan T. Black. "Mathematical Relation between the Hohmann Transfer and Continuous-Low Thrust Maneuvers," *Acta Astronautica*, 96: 42-44 (2014).
- Brown, Charles D. *Elements of Spacecraft Design*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2002.
- Brunner, Christopher W. and Ping Lu. "Skip Entry Trajectory Planning and Guidance," *Journal of Guidance, Control, and Dynamics* 31, no. 5: 1210-1219 (2008).
- Busemann, Adolf, Nguyen X. Vinh, and Robert D. Culp. "Solution of the Exact Equations for Three-Dimensional Atmospheric Entry Using Directly Matched Asymptotic Expansions," *NASA CR-2643*. Washington, D.C.: National Aeronautics and Space Administration, 1976.
- Capderou, Michel. *Satellites: Orbits and Missions*. Paris, France: Springer-Verlag France, 2005.
- Chapman, Dean R. "An Approximate Analytical Method for Studying Entry into Planetary Atmospheres," *NACA TN 4276*. Moffett Field, CA: AMES Aeronautical Laboratory, 1958.
- Chesley, Bruce, Reinhold Lutz, and Robert F. Brodsky. "Space Payload Design and Sizing," in *Space Mission Analysis and Design*. Ed. James R. Wertz and Wiley J. Larson. El Segundo, CA: Microcosm Press, 2003.
- Chobotov, Vladimir A. *Orbital Mechanics*, Third Edition. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2002.
- Christol, Carl Q. "The Aerospace Plane: Its' Legal and Political Future," *Space Policy* 9, no. 1: 35-43 (1993).
- Co, Thomas C. "Operationally Responsive Spacecraft Using Electric Propulsion." Ph.D Dissertation, AFIT-ENY-DS-12-01. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, September 2012 (ADA564646).
- Committee on the Peaceful Uses of Outer Space. *Questionnaire on Possible Legal Issues with Regard to Aerospace Objects: Replies from Member States*. UN doc. A/AC.105/635. Vienna, Austria: United Nations Office at Vienna, 15 February 1996.
- Dalton, Devin K. "Ground Target Over-Flight and Orbital Maneuvering via Aeroassisted Maneuvers." MS Thesis, AFIT-ENY-14-M-12. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2014.
- Darby, Christopher L., and Anil V. Rao. "Optimal Impulsive LEO to LEO Multiple-Pass Aeroassisted Orbital Transfer for Small Spacecraft." Paper presented at the *20th AAS/AIAA Space Flight Mechanics Meeting*, San Diego, CA, 14-17 February 2010.

- Darby, Christopher. L., and Anil V. Rao. "Minimum-Fuel Low-Earth-Orbit Aeroassisted Orbital Transfer of Small Spacecraft," *Journal of Spacecraft and Rockets* 48, no. 4: 618-628 (2011).
- Daryabeigi, Kamran. "Thermal Analysis and Design of Multilayer Insulation for Re-Entry Aerodynamic Heating," *Journal of Spacecraft and Rockets* 39, no. 4: 509-514 (2002).
- Der, Gim J. "The Superior Lambert Algorithm." Paper presented at the *Advanced Maui Optical and Space Surveillance Technologies Conference*, Wailea, Maui, HI, 13-16 September 2011.
- Detra, R. W., N.H. Kemp, and F. R. Riddell. "Addendum to 'Heat Transfer to Satellite Vehicles Re-Entering the Atmosphere,'" *Jet Propulsion* 27: 1256-1257 (1957).
- Deza, Michel M. and Elena Deza. *Encyclopedia of Distances*, Second Edition. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2013.
- Diederiks-Verschoor, I. H. Ph. and V. Kopal. *An Introduction to Space Law*, Third Edition. Alphen aan den Rijn, The Netherlands: Kluwer Law International, 2008.
- Eggers Jr., A. J. and Thomas J. Wong. "Motion and Heating of Lifting Vehicles during Atmosphere Entry," *American Rocket Society (ARS) Journal*, 31: 1364-1375 (1961).
- Epperson, James F. *An Introduction to Numerical Methods and Analysis*. Hoboken, NJ: John Wiley & Sons, Inc., 2007.
- Galman, Barry A. "Some Fundamental Considerations for Lifting Vehicles in Return from Satellite Orbit," *Planetary and Space Science*, 4: 399-410 (1961).
- Gargasz, Michael L. "Optimal Spacecraft Attitude Control Using Aerodynamic Torques." MS Thesis, AFIT-ENY-GA-07-M-08. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2007 (ADA469919).
- Geddes, Keith O., Stephen R. Czapor, George Labahn. *Algorithms for Computer Algebra*. Norwell, MA: Kluwer Academic Publishers, 1992.
- Giunta, Anthony A., Vladimir Balabanov, Dan Haim, Bernard Grossman, William H. Mason, Layne T. Watson, and Raphael T. Haftka. "Multidisciplinary Optimization of a Supersonic Transport Using Design of Experiments Theory and Response Surface Modeling," *TR 97-10*. Blacksburg, VA: Virginia Polytechnic Institute and State University, 2001.
- Gong, Leslie, William L. Ko, Robert D. Quinn, and W. Lance Richards. "Comparison of Flight-Measured and Calculated Temperatures on the Space Shuttle Orbiter," *NASA TM 88278*. Edwards, CA: NASA Dryden Flight Research Facility, 1987.

- Gonzalez, Daniel, Mel Eisman, Calvin Shipbaugh, Timothy Bonds, and Anh Tuan Le. *Proceedings of the RAND Project AIR FORCE Workshop on Transatmospheric Vehicles*. Santa Monica, CA: RAND Corporation, 1997.
- Gooding, R. H. "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem," *Celestial Mechanics and Dynamical Astronomy* 48, no. 2: 145-165 (1990).
- Gorove, Katherine M. "Delimitation of Outer Space and the Aerospace Object – Where is the Law?" *Journal of Space Law* 28, no. 1: 11-28 (2000).
- Granger, Robert A. *Fluid Mechanics*. Mineola, NY: Dover Publications, Inc., 1995.
- Guettler, David B. "Satellite Attitude Control Using Atmospheric Drag." MS Thesis, AFIT-ENY-GA-07-M-10. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2007 (ADA469265).
- Hajovsky, Blake B. "Satellite Formation Control Using Atmospheric Drag." MS Thesis, AFIT-ENY-GA-07-M-11. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2007 (ADA469289).
- Hall, Timothy S. "Orbit Maneuver for Responsive Coverage Using Electric Propulsion." MS Thesis, AFIT-ENY-GSS-10-M-04. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2010 (ADA516854).
- Hanson, John M. "Combining Propulsive and Aerodynamic Maneuvers to Achieve Optimal Orbital Transfer," *Journal of Guidance, Control, and Dynamics* 12, no. 5: 732-738 (1989).
- Harris, Alexandra and Ray Harris. "The Need for Air Space and Outer Space Demarcation," *Space Policy* 22, no. 1: 3-7 (2006).
- Havey Jr., Keith A. "Entry Vehicle Performance in Low-Heat-Load Trajectories," *Journal of Spacecraft and Rockets*, 19: 506-512 (1982).
- Hedayat, A. S., N. J. A. Sloane, and John Stufken. *Orthogonal Arrays: Theory and Applications*. New York, NY: Springer Verlag New York, Inc., 1999.
- Heppenheimer, T. A. *The Space Shuttle Decision: NASA's Search for a Reusable Space Vehicle*. Washington, D.C.: National Aeronautics and Space Administration, 1999.
- Hicks, Kerry D. *Introduction to Astrodynamic Re-Entry*, TR 09-03. Wright-Patterson AFB, OH: Air Force Institute of Technology, 2009.
- Hobe, Stephan, Gerardine M. Goh, and Julia Neumann. "Space Tourism Activities – Emerging Challenges to Air and Space Law?" *Journal of Space Law* 33, no. 2: 359-374 (2007).

- Huttenlocher, Daniel P., Gregory A. Klanderman, and William J. Rucklidge. "Comparing Images Using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, no. 9: 850-863 (1993).
- Ikawa, H. and T. F. Rudiger. "Synergetic Maneuvering of Winged Spacecraft for Orbital Plane Change." Paper presented at the *AIAA 20th Aerospace Sciences Meeting*, Orlando, FL, 11-14 January 1982.
- Jolley, Patrick R. and Stephen A. Whitmore. "Aerodynamic and Propulsion Assisted Maneuvering for Orbital Transfer Vehicles." Paper presented at the *5th Responsive Space Conference*, Los Angeles, CA, 23-26 April 2007.
- Johnson, Richard E. "Effects of Thrust Vector Control on the Performance of the Aerobang Orbital Plane Change Maneuver." MS Thesis; Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, Monterey, CA, June 1993 (ADA272532).
- Kay, W. D. "The X-15 Hypersonic Flight Research Program: Politics and Permutations at NASA," in *From Engineering Science to Big Science: The NACA and NASA Collier Trophy Research Project Winners*. Ed. Pamela E. Mack. Washington D.C.: U.S. Government Printing Office, 1998.
- Kelley, C. T. *Solving Nonlinear Equations with Newton's Method*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2003.
- King-Hele, Desmond. *Satellite Orbits in an Atmosphere: Theory and Applications*. Glasgow, Scotland: Blackie and Son Ltd., 1987.
- Kleijnen, Jack P. C. *Design and Analysis of Simulation Experiments*. New York, NY: Springer Science + Business Media, LLC, 2008.
- Kluever, C. A. "Entry Guidance Using Analytical Atmospheric Skip Trajectories," *Journal of Guidance, Control, and Dynamics* 31, no. 5: 1531-1534 (2008).
- Ko, William L., Leslie Gong, and Robert D. Quinn. "Re-Entry Thermal Analysis of a Generic Crew Exploration Vehicle Structure," *NASA TM 2007-214607*. Edwards, CA: NASA Dryden Flight Research Facility, 2007.
- Ko, William L., Robert D. Quinn, and Leslie Gong. "Finite Element Re-Entry Heat Transfer Analysis of Space Shuttle Orbiter," *NASA TP 2657*. Edwards, CA: NASA Dryden Flight Research Facility, 1986.
- Kovvali, Narayan. *Theory and Applications of Gaussian Quadrature Methods*. New York: Morgan & Claypool Publishers, 2011.

- Kuneš, Joseph. *Dimensionless Physical Quantities in Science and Engineering*. Waltham, MA: Elsevier Inc., 2012.
- Lee, Dorothy B., John J. Bertin, and Winston D. Goodrich. "Heat-Transfer Rate and Pressure Measurements Obtained during Apollo Orbital Entries," *NASA TN D-6028*. Washington, D.C.: National Aeronautics and Space Administration, 1970.
- Levin, Henry M. and Patrick J. McEwan. *Cost-Effectiveness Analysis*, Second Edition. Thousand Oaks, CA: Sage Publications, Inc., 2001.
- Lindeburg, Michael P. *Engineering Unit Conversion*, Fourth Edition. Belmont, CA: Professional Publications, Inc., 1999.
- Long, S.A.T. "General-Altitude Transformations between Geocentric and Geodetic Coordinates," *Celestial Mechanics* 12: 225-230 (1975).
- Longley, Paul, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. Hoboken, NJ: John Wiley & Sons, Inc., 2005.
- Lundstedt, Torbjörn, Elisabeth Seifert, Lisbeth Abramo, Bernt Thelin, B., Åsa Nyström, Jarle Petterson, and Rolf Bergman. "Experimental Design and Optimization," *Chemometrics and Intelligent Laboratory Systems* 42, no. 1-2: 3-40 (1998).
- Lyall, Francis and Paul B. Larsen. *Space Law: A Treatise*. Surrey, United Kingdom: Ashgate Publishing Limited, 2009.
- Marín, E., A. Calderón, and O. Delgado-Vasallo. "Similarity Theory and Dimensionless Numbers in Heat Transfer," *European Journal of Physics* 30: 439-445 (2009).
- McNabb, Dennis J. "Investigation of Atmospheric Re-Entry for the Space Maneuver Vehicle." MS Thesis, AFIT-ENY-GA-04-M-03. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2004 (ADA424074).
- McNish, Larry. "Latitude and Longitude." RASC Calgary Centre, The Royal Astronomical Society of Canada. Last modified 11 November 2011. Accessed 17 August 2012. <http://calgary.rasc.ca/latlong.htm>.
- Miele, A., W. Y. Lee, and K. D. Mease. "Optimal Trajectories for LEO-to-LEO Aeroassisted Orbital Transfer," *Acta Astronautica* 18: 99-122 (1988).
- Moore, F. K. "Entry Radiative Transfer," in *Re-Entry and Planetary Entry Physics and Technology: Dynamics, Physics, Radiation, Heat Transfer, and Ablation*. Ed. W. H. T. Loh. New York, NY: Springer-Verlag New York Inc., 1968.

- Murphy, Kevin J., Robert J. Nowak, Richard A. Thompson, and Brian R. Hollis. "X-33 Hypersonic Aerodynamic Characteristics," *Journal of Spacecraft and Rockets* 38, no. 5: 670-683 (2001).
- Naidu, D. S. "Three-Dimensional Atmospheric Entry Problem Using Method of Matched Asymptotic Expansions," *IEEE Transactions on Aerospace and Electronic Systems* 25, no. 5: 660-667 (1989).
- National Aeronautics and Space Administration. "Process for Limiting Orbital Debris," *NASA STD 8719.14A*. Washington, D.C.: National Aeronautics and Space Administration, 2012.
- National Research Council. Aeronautics and Space Engineering Board. *Continuing Kepler's Quest: Assessing Air Force Space Command's Astrodynamics Standards*. Washington, D.C.: The National Academies Press, 2012.
- Newberry, Robert D. "Powered Spaceflight for Responsive Space Systems," *High Frontier* 1: 46-49 (2005).
- Nicholson, John C. "Numerical Optimization of Synergistic Maneuvers." MS Thesis; Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, Monterey, CA, June 1994 (ADA283398).
- O'Leary, Michael. *Revolutions of Geometry*. Hoboken, NJ: John Wiley & Sons, Inc., 2010.
- Olfe, Daniel B. "Radiation Gasdynamics," in *Radiation and Re-Entry*. Ed. S. S. Penner and Daniel B. Olfe. New York, NY: Academic Press Inc., 1968.
- Parish II, Michael S. "Optimality of Aeroassisted Orbital Plane Changes." MS Thesis; Department of Aeronautical and Astronautical Engineering, Naval Postgraduate School, Monterey, CA, December 1995 (ADA306573).
- Peterson, Roger P. *Design and Analysis of Experiments*. New York, NY: Marcel Dekker, Inc., 1985.
- Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge, United Kingdom: Cambridge University Press, 1988.
- Powell, R.W., J.C. Naftel, and M.J. Cunningham. "Performance Evaluation of an Entry Research Vehicle," *Journal of Spacecraft and Rockets* 24: 489-495 (1987).
- Putnam, Zachary R., Matthew D. Neave, and Gregg H. Barton. "PredGuid Entry Guidance for Orion Return from Low Earth Orbit." Paper presented at the *2010 IEEE Aerospace Conference*, Big Sky, Montana, 6-13 March 2010.

- Rao, Anil V. and Arthur E. Scherich. "A Concept for Operationally Responsive Space Mission Planning Using Aeroassisted Orbital Transfer." Paper presented at the *6th Responsive Space Conference*, Los Angeles, CA, 28 April – 1 May 2008.
- Rao, Anil V., Sean Tang, and Wayne P. Hallman. "Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer," *Optimal Control Applications and Methods* 23: 215-238 (2002).
- Rekab, Kamel and Muzaffar Shaikh. *Statistical Design of Experiments with Engineering Applications*. Boca Raton, FL: Taylor and Francis Group, LLC., 2005.
- Ross, I. Michael and John C. Nicholson. "Optimality of the Heating-Rate-Constrained Aerocruise Maneuver," *Journal of Spacecraft and Rockets* 35, no. 3: 361-364 (1998).
- Silva, J. Dario Landa, Edmund K. Burke, and Sanja Petrovic. "An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling," in *Metaheuristics for Multiobjective Optimization*. Ed. Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen, and Vincent T'kindt. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2004.
- Speyer, J. L. and M. E. Womble. "Approximate Optimal Atmospheric Entry Trajectories," *Journal of Spacecraft and Rockets* 8: 1120-1125 (1971).
- Stanik, Joseph T. *El Dorado Canyon: Reagan's Undeclared War with Qaddafi*. Annapolis, MD: Naval Institute Press, 2003.
- Storch, J. A. *Aerodynamic Disturbances on Spacecraft in Free-Molecular Flow*. El Segundo, CA: The Aerospace Corporation, 2002.
- Su, Jinyuan. "Near Space as a *Sui Generis* Zone: A Tri-Layer Approach of Delimitation," *Space Policy* 29, no. 2: 90-92 (2013).
- Sun, Yong and Maorui Zhang. "Optimal Re-Entry Range Trajectory of Hypersonic Vehicle by Gauss Pseudospectral Method." Paper presented at the *2nd International Conference on Intelligent Control and Information Processing*, Harbin, China, 25-28 July 2011.
- Talbi, El-Ghazali. *Metaheuristics: From Design to Implementation*. Hoboken, NY: John Wiley & Sons, Inc., 2009.
- Tauber, Michael E. "Maximum Lift/Drag Ratio of Flat Plates with Bluntness and Skin Friction at Hypersonic Speeds," *NASA TM 88338*. Moffett Field, CA: AMES Research Center, 1986.
- Terekhov, Andrei D. "Passage of Space Objects through Foreign Airspace: International Custom?" *Journal of Space Law* 25, no. 1: 1-16 (1997).
- Tewari, Ashish. *Atmospheric and Space Flight Dynamics*. Boston, MA: Birkhäuser, 2007.

- The Center for Space Standards & Innovation. “Molniya 3-42 TLE.” NORAD Two-Line Element Sets, CelesTrak. Last modified 29 January 2014. Accessed 29 January 2014. <http://www.celestrak.com/NORAD/elements/molniya.txt>.
- United Nations General Assembly. *Convention on Registration of Objects Launched into Outer Space*. 28 U.S.T. 695, 1023 U.N.T.S. 15. New York, NY: United Nations Office at New York, 14 January 1975.
- Vallado, David A. *Fundamentals of Astrodynamics and Applications*, Third Edition. El Segundo, CA: Microcosm Press, 2007.
- Van Nimmen, Jane, Leonard C. Bruno, and Linda N. Ezell. *NASA Historical Data Book, Volume VII: NASA Launch Systems, Space Transportation, Human Spaceflight, and Space Science, 1989-1998*. Washington D.C.: U.S. Government Printing Office, 1999.
- Viikari, Lotta. *The Environmental Element in Space Law: Assessing the Present and Charting the Future*. Leiden, The Netherlands: Koninklijke Brill NV, 2008.
- Vincenty, Thaddeus. “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Applications of Nested Equations,” *Survey Review* XXII 176: 88-93 (1975).
- Vinh, Nguyen X., Adolf Busemann, and Robert D. Culp. *Hypersonic and Planetary Entry Flight Mechanics*. Ann Arbor, MI: The University of Michigan Press, 1980.
- Vinh, N. X., A. Busemann, and R. D. Culp. “Optimum Three-Dimensional Atmospheric Entry,” *Acta Astronautica* 2: 593-611 (1975).
- Vinh, N. X. and John M. Hanson. “Optimal Aeroassisted Return from High Earth Orbit with Plane Change,” *Acta Astronautica*, 12: 11-25 (1985).
- Vinh, N. X. and Der-Ming Ma. “Optimal Multiple-Pass Aeroassisted Plane Change,” *Acta Astronautica* 21: 749-758 (1990).
- Vinh, N. X. and Ya-Wen Shih. “Optimum Multiple-Skip Trajectories,” *Acta Astronautica* 41: 103-112 (1997).
- Wang, Yalin, Qilong Han, and Haiwei Pan. “A Clustering Scheme for Trajectories in Road Networks,” in *Advanced Technology in Teaching – Proceedings of the 2009 3rd International Conference on Teaching and Computational Science*. Ed. Yanwen Wu. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2012.
- Weisstein, Eric W. *CRC Concise Encyclopedia of Mathematics*, Second Edition. Boca Raton, FL: CRC Press LLC, 2003.

Vita

Captain Robert A. Bettinger graduated from Niagara-Wheatfield Central High School in Wheatfield, New York in June 2003. He attained a nomination to and entered the United States Air Force Academy in Colorado Springs, Colorado where he graduated with a Bachelor of Science in Astronautical Engineering, was recognized as a Distinguished Graduate, and received a Regular Commission in May 2007.

As his first assignment, Captain Bettinger was sent to Kirtland AFB, New Mexico where he began work as a Research Engineer for Spacecraft Navigation and Guidance in the Air Force Research Laboratory's Space Vehicle Directorate in November 2007. Soon after his start at Kirtland AFB, Captain Bettinger entered the American Public University (APU) and, in May 2010, attained his Master's degree in History with an emphasis on European History. Following his assignment in New Mexico, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology (AFIT) in May 2010 and completed his Master's degree in Astronautical Engineering in June 2011. Immediately thereafter, Captain Bettinger transitioned into the Ph.D program at AFIT and continued his graduate research in re-entry dynamics and aeroassisted, trans-atmospheric maneuvers. Following an interim assignment at the National Air and Space Intelligence Center (NASIC) upon conferral of his doctorate in 2014, Captain Bettinger will be assigned to the United States Air Force Academy in 2017 where he will be an instructor within the Department of Astronautics.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.				
1. REPORT DATE (DD-MM-YYYY) 19-06-2014		2. REPORT TYPE Dissertation		3. DATES COVERED (From — To) July 2011 – June 2014
4. TITLE AND SUBTITLE The Prospect of Responsive Spacecraft Using Aeroassisted, Trans-Atmospheric Maneuvers		5a. CONTRACT NUMBER		
		5b. GRANT NUMBER		
		5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Bettinger, Robert A, Capt, USAF		5d. PROJECT NUMBER		
		5e. TASK NUMBER		
		5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management 2950 Hobson Way WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENY-DS-14-J-13		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank		10. SPONSOR/MONITOR'S ACRONYM(S) N/A		
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A		
12. DISTRIBUTION / AVAILABILITY STATEMENT Distribution Statement A. Approved for Public Release; Distribution Unlimited				
13. SUPPLEMENTARY NOTES This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.				
14. ABSTRACT Comprised of exo- and trans-atmospheric trajectory segments, atmospheric re-entry represents a complex dynamical event which traditionally signals the mission end-of-life for low-Earth orbit spacecraft. Transcending this paradigm, atmospheric re-entry can be employed as a means of operational maneuver whereby aerodynamic forces can be exploited to create an aeroassisted maneuver. Utilizing a notional trans-atmospheric, lifting re-entry vehicle with L/D =6, the first phase of research demonstrates the terrestrial reachability potential for skip entry aeroassisted maneuvers. By overflying a geographically diverse set of ground targets, comparative analysis indicates a significant savings in ΔV expenditure for skip entry compared with exo-atmospheric maneuvers. In the second phase, the Design of Experiments method of orthogonal arrays provides optimal vehicle and skip entry trajectory designs by employing main effects and Pareto front analysis. Depending on re-circularization altitude, the coupled optimal design can achieve an inclination change of 19.91 deg with 50-85% less ΔV than a simple plane change. Finally, the third phase introduces the descent-boost aeroassisted maneuver as an alternative to combined Hohmann and bi-elliptic transfers in order to perform LEO injection. Compared with bi-elliptic transfers, simulations demonstrate that a lifting re-entry vehicle performing a descent-boost maneuver requires 6-12% less for injection into orbits lower than 650 km. In addition, the third phase also introduces the "Maneuver Performance Number" as a dimensionless means of comparative maneuver effectiveness analysis.				
15. SUBJECT TERMS Aeroassisted; re-entry; skip entry; descent-boost; plane change; design of experiments; MP number				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 413
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U		
			19a. NAME OF RESPONSIBLE PERSON Dr. Jonathan T. Black, AFIT/ENY	
			19b. TELEPHONE NUMBER (Include Area Code) (937) 255-6565 x4578 jonathan.black@afit.edu	